คู่มือการใช้ไทยปกาลิต (ภาษาเบสิก คำสั่งไทย)

001.6424
ข 136 ค

คู่มือการใช้ไทยปกาสิต (ภาษาเบสิก คำสั่งไทย)
ขจรศักดิ์ คันธพนิต

โปรดทราบ

แผ่นวงจรการ์ด "ไทยปกาสิต" สำหรับเสียบบนเครื่อง Apple II และ Apple II Compatible ใกล้จะออกวางตลาดแล้ว ผู้สนใจ โปรดติดต่อสอบถามรายละเอียดได้ที่

สำนักที่ปรึกษาคอมพิวเตอร์เชียงใหม่
81 ถนนราชวิถี อำเภอเมือง
เชียงใหม่ 50000
โทร. (053) 214482

10508

พิมพ์ครั้งที่ 1 จำนวน 200 เล่ม
ISBN 974-87349-3-5
ราคา 60 บาท
สงวนลิขสิทธิ์

เลขห้อง REBK
เลขหมู่ 001.6424 ข136ค
เลขทะเบียน
NATIONAL LIBRARY OF THAILAND
31111016407015

คำนำ

คอมพิวเตอร์ กำลังจะกลายเป็น เครื่องใช้สำนักงานที่จำเป็น อีกอย่างหนึ่ง แต่ในประเทศไทยของเรา ยังประสบปัญหาอยู่มาก ในการผลิตบุคคลากรให้มีความสามารถใช้คอมพิวเตอร์ ได้อย่างมีประสิทธิภาพ สาเหตุสำคัญประการหนึ่งคือ การเรียนคอมพิวเตอร์เริ่มที่อายุมากเกินไป หากเราสามารถทำให้เด็ก ๆ ของเรา เริ่มเรียนคอมพิวเตอร์ได้ ตั้งแต่อายุ 7-10 ขวบแล้ว ปัญหาด้านการขาดแคลนบุคคลากรคอมพิวเตอร์ในอนาคต จะไม่เกิดขึ้นอย่างแน่นอน แต่เด็ก ๆ ของเราส่วนใหญ่ ไม่ได้เรียนภาษาอังกฤษจนกระทั่งอายุ 11-12 ปี และกว่าจะรู้ภาษาอังกฤษมากพอที่จะเริ่มเรียนคอมพิวเตอร์ได้ ก็ต้องรอถึงอายุ 15-16 ปี

ด้วยความปรารถนาอย่างแรงกล้าที่จะหาทางเอาชนะปัญหานี้ คณะผู้วิจัยจึงได้ทำการค้นคว้าทดลอง จนในที่สุด สามารถสร้างระบบคอมพิวเตอร์ที่ใช้คำสั่งเป็นภาษาไทยทั้งหมด และสร้างภาษาเขียนโปรแกรมด้วยคำสั่งไทยขึ้น โดยใช้ชื่อว่า "ไทยปกาสิต"

หนังสือเล่มนี้ เป็นคู่มือสำหรับใช้เรียน วิธีเขียนโปรแกรมด้วยภาษา "ไทยปกาสิต" โดยที่ผู้เริ่มเรียนที่ไม่มีพื้นความรู้มาก่อนเลยก็สามารถใช้ได้ แม้ว่าในตอนแรก "ไทยปกาสิต" จะใช้ได้เฉพาะบนเครื่องคอมพิวเตอร์ "ลำไย 28" ซึ่งเป็นเครื่องต้นแบบที่ได้รับการออกแบบเป็นพิเศษเท่านั้น แต่คณะผู้วิจัยเชื่อว่า ในไม่ช้าเครื่องคอมพิวเตอร์หลาย ๆ ยี่ห้อ ที่มีจำหน่ายในประเทศไทยจะต้องได้รับการพัฒนาให้สามารถใช้ "ไทยปกาสิต" ได้ด้วย เมื่อถึงวันนั้น เด็กไทยจะสามารถเริ่มเรียนคอมพิวเตอร์ได้ ทันทีที่อ่านออกและเขียนภาษาไทยได้

อนึ่ง คณะผู้วิจัยกำลังออกแบบสร้างแผงวงจร เป็นการ์ด สำหรับเสียบบนเครื่อง Apple II หรือ Apple II Compatible ซึ่งจะทำให้เครื่องดังกล่าวที่มีอยู่อย่างดาษดื่น สามารถใช้ "ไทยปกาสิต" ได้

คณะผู้วิจัย มุ่งหวังที่จะให้หนังสือเล่มนี้ เป็นสื่อช่วยให้บรรดาครูบาอาจารย์ ตลอดจนผู้สนใจทั้งหลาย ได้มองเห็นความสำคัญ ของการมีภาษาเขียนโปรแกรมที่เป็นภาษาไทย และมองเห็นความง่ายในการเรียน วิธีเขียนโปรแกรมด้วยภาษา "ไทยปกาสิต" เพื่อจะได้ช่วยกันส่งเสริมให้มีการใช้กันอย่างแพร่หลายต่อไป ทั้งนี้ เพื่อให้บังเกิดผลดีต่อการศึกษาของเด็ก ๆ ซึ่งเป็นอนาคตของชาติโดยแท้

คณะผู้วิจัย

1 กค 2528

สารบาญ

หน้า

1. คอมพิวเตอร์เพื่อคนไทย 1

1.1 คอมพิวเตอร์ 1

1.2 ข้อมูล 1

1.3 ฮาร์ดแวร์กับซอฟท์แวร์ 2

1.4 คำสั่งคอมพิวเตอร์ 2

1.5 ภาษาเบสิค 3

1.6 คอมพิวเตอร์กับภาษาไทย 4

1.7 คอมพิวเตอร์คำสั่งไทย "ลำไย 28" 5

1.8 ภาษาเบสิคคำสั่งไทย "ไทยปกาสิต" 5

1.9 ของคนไทย โดยคนไทย และเพื่อคนไทย 6

2. วิธีใช้เครื่องคอมพิวเตอร์ "ลำไย 28" 7

2.1 คุณสมบัติพิเศษของ "ลำไย 28" 7

2.2 แป้นพิมพ์ 7

2.3 การเปลี่ยนภาษาของแป้นพิมพ์ 10

2.4 การเลือกระบบแปลภาษา 10

2.5 การแสดงตัวอักษรบนจอภาพ 11

2.6 การแปลคำสั่งจาก อังกฤษเป็นไทย หรือจากไทยเป็นอังกฤษ 11

2.7 การเลื่อนเคอร์เซอร์ 12

3. คำแนะนำเบื้องต้นในการใช้ไทยปกาสิต 13

3.1 สำหรับผู้ที่รู้ภาษาเบสิคแล้ว 13

3.2 คำแนะนำเบื้องต้นสำหรับผู้เริ่มเรียน 13

3.3 คำสั่งตรงกับคำสั่งในโปรแกรม 13

3.4 เลขบรรทัดกับเครื่องหมายวรรคตอน 15

3.5 การคำนวนและผลลัพธ์ 17

3.6 สัญญาณบอกข้อผิดพลาด 18

3.7 การแจง 18

3.8 การแก้ไขโปรแกรม 19

3.9 จำนวนเลข 20

หน้า
3.10 การแสดงจำนวนเลขบนจอ 21
3.11 ตัวแปร 22
3.12 ค่าของตัวแปร 24
4. การเขียนโปรแกรมด้วย "ไทยปกาสิต" 27
4.1 สิ่งที่จะต้องรู้ในการเขียนโปรแกรม 27
4.2 พิมพ์ 27
4.3 การกำหนดค่าตัวแปร 28
4.4 การใช้หมายเหตุ 29
4.5 รับ และ จับ 29
4.6 ถ้า...แล้ว และ ไป 31
4.7 จาก...เพิ่ม 32
4.8 ไปภาค และ กลับ 33
4.9 การใช้คำสั่ง จาก...เพิ่ม ซ้อนกันหลายชั้น 35
4.10 การใช้คำสั่ง ไปภาค ซ้อนกันหลายชั้น 36
4.11 ขึ้นกับ A ... ไป และ ขึ้นกับ A ... ไปภาค 36
4.12 อ่าน และ ข้อมูล 37
4.13 สงวน 39
4.14 ตัวแปรแถว 39
4.15 ฟังก์ชั่นของตัวแปรอักขระ 40
4.16 การจัดรูปแบบบนจอ 41
4.17 สรุป 42
5. ประมวลคำสั่งของไทยปกาสิต 44
5.1 ข้อตกลงเกี่ยวกับสัญญลักษณ์ 44
5.2 คำสั่งทำการ (เรียงตามลำดับอักษร) 44
5.3 ฟังก์ชั่น (เรียงตามลำดับอักษร) 72
ภาคผนวก
ผ.1 คำสงวนของไทยปกาสิต 80
ผ.2 ตารางรหัสอักษร 81
ผ.3 รายงานความผิดพลาด 83

บทที่ 1

คอมพิวเตอร์ไทย เพื่อคนไทย

1.1 คอมพิวเตอร์

คอมพิวเตอร์ คือ เครื่องอิเล็กทรอนิกส์ชนิดหนึ่ง ซึ่งสามารถทำงานได้โดย **อัตโนมัติ** ตามคำสั่งที่เรียบเรียงและบันทึกไว้ในหน่วย**ความจำ** และสามารถทำการ**ตัดสินใจ** ได้ตามเงื่อนไขที่กำหนดในคำสั่ง คำที่พิมพ์ตัวเข้มไว้ทั้งสามคำในข้อความข้างต้นนั้น เป็นคำกำหนดคุณสมบัติของคอมพิวเตอร์ ซึ่งจะต้องมีครบทั้ง 3 ประการ หากขาดประการใดประการหนึ่ง ก็ไม่นับว่าเป็นคอมพิวเตอร์ ยกตัวอย่างเช่น เครื่องคำนวณ (เครื่องคิดเลข) แบบโปรแกรมได้ (Programmable Calculator) มีคุณสมบัติเพียง 2 ข้อ คือ ทำงานได้โดยอัตโนมัติ และมีความจำ แต่ไม่สามารถตัดสินใจได้ จึงไม่ถือว่าเป็นคอมพิวเตอร์

เราอาจสรุปได้ว่า คอมพิวเตอร์มีคุณสมบัติ 3 ประการ คือ

1. ทำงานได้โดย**อัตโนมัติ** (ตามคำสั่งที่เรียบเรียงไว้ล่วงหน้า)
2. มี **ความจำ** (สำหรับเก็บคำสั่ง และข้อมูล)
3. สามารถ **ตัดสินใจ** ได้ (ตามเงื่อนไขที่กำหนดในคำสั่ง)

1.2 ข้อมูล

ข้อมูล คือ **ตัวเลข** หรือ **ข้อความ** ซึ่งคอมพิวเตอร์สามารถนำไป **จัดการ** ได้

การจัดการกับตัวเลข ได้แก่การ บวก ลบ คูณ หาร ยกกำลัง หรือคำนวณตามสูตรต่างๆ ในทางคณิตศาสตร์

การจัดการกับข้อความ ได้แก่การ ตัดคำ ต่อคำ เปรียบเทียบ และค้นหาคำ เป็นต้น

ขบวนการจัดการกับข้อมูลของคอมพิวเตอร์นั้น อาศัยหลัก 2 ประการคือ

1. การเข้ารหัส ตัวเลข และตัวอักษรทุกตัว จะถูกแทนด้วยรหัสเลขฐานสอง (ซึ่งประกอบด้วยเลข 0 กับเลข 1 เท่านั้น) เลขฐานสอง 1 หลัก เรียกว่า 1 บิท (BIT ย่อมาจาก binary digit) หน่วยของข้อมูลเรียกว่า ไบท์ (bite) ประกอบด้วยเลขฐานสอง 8 หลัก

ตัวอย่าง เช่น รหัสซึ่งใช้แทนอักษร A คือ 01000001 เป็นข้อมูล 1 ไบท์ เลขจำนวนเต็มพร้อมเครื่องหมายแทนด้วยข้อมูล 2 ไบท์ เป็นต้น

2. การคำนวณทางคณิตศาสตร์และตรรก ในเมื่อข้อมูลทุกอย่างถูกเข้ารหัสเป็นตัวเลขดังกล่าวแล้ว การจัดการกับข้อมูลจึงเป็นการคำนวณเกี่ยวกับตัวเลขในระบบเลขฐานสอง การคำนวณต่างๆ อาศัยหลักของเลขคณิต (Arithmetic) และตรรกศาสตร์ (Logic) ซึ่งทำให้คอมพิวเตอร์มีความสามารถในการคำนวณและการตัดสินใจ

1.3 ฮาร์ดแวร์ กับ ซอฟท์แวร์

คุณสมบัติของคอมพิวเตอร์ตามข้อ 1.1 และหลักการจัดการกับข้อมูลตามข้อ 1.2 นั้น มาจากวงจรอิเล็คทรอนิคส์ ที่ได้รับการออกแบบเป็นพิเศษ วงจรดังกล่าว และส่วนประกอบอื่นๆ เช่น จอภาพ แป้นพิมพ์ ฯลฯ เราเรียกว่า ฮาร์ดแวร์ (Hardware)

แต่การทำงานของคอมพิวเตอร์นั้นจะต้องอาศัยส่วนประกอบที่สำคัญอีกอย่างหนึ่ง คือ คำสั่งที่สั่งให้เครื่องทำงาน ซึ่งแบ่งออกได้เป็นสองประเภท

คำสั่งประเภทแรกเป็นคำสั่งภายใน เกี่ยวกับการควบคุมการทำงานของฮาร์ดแวร์ต่างๆ คำสั่งประเภทนี้เรียกว่า เฟิร์มแวร์ (FIRMWARE) แต่ในทัศนะของผู้ใช้คอมพิวเตอร์อาจถือว่าเป็นส่วนหนึ่งของฮาร์ดแวร์ก็ได้

คำสั่งประเภทที่สองคอคำสั่งภายนอกที่ผู้ใช้คอมพิวเตอร์สามารถใช้สั่งให้เครื่องทำงานตามที่ต้องการ คำสั่งประเภทนี้มีชื่อเรียกว่า ซอฟท์แวร์ (SOFTWARE)

1.4 คำสั่งคอมพิวเตอร์

คำสั่งที่จะให้คอมพิวเตอร์สามารถเข้าใจและปฏิบัติตามได้นั้น ต้องอยู่ในรูปของรหัสเลขฐานสอง เช่นเดียวกับข้อมูล แต่การที่จะเขียนคำสั่งด้วยรหัสเลขฐานสองนั้น เป็นภาระอันหนักสำหรับคน เพราะคนไม่สามารถอ่านรหัสเลขฐานสองให้เข้าใจได้ทันที จำเป็นต้องเปิดตารางและเทียบทีละคำสั่ง วิธีนี้นอกจากจะเสียเวลามากแล้ว ยังเกิดความผิดพลาดได้ง่ายอีกด้วย จึงได้มีการพัฒนาภาษาสำหรับเขียนคำสั่งคอมพิวเตอร์ให้มีลักษณะพิเศษ คือ คนสามารถอ่านเข้าใจได้ แต่ในขณะเดียวกันก็มีกฏเกณฑ์อย่างรัดกุมพอที่เครื่องคอมพิวเตอร์จะนำไปแปลเป็นรหัสเลขฐานสองได้

โปรแกรม (Program) คือชุดคำสั่งที่เรียบเรียงไว้ให้คอมพิวเตอร์ทำงานตามที่เราต้องการ โปรแกรมอาจเขียนด้วยรหัสเลขฐานสองดังกล่าวแล้วข้างต้น ซึ่งเรียกว่า เขียนเป็น**ภาษาเครื่อง** (machine language) หรืออาจเขียนด้วย

ภาษาเขียนโปรแกรม (Programming language) ภาษาใดภาษาหนึ่ง

ภาษาเขียนโปรแกรมระดับต่ำสุด คือ **ภาษาแอสเซมบลี** (Assembly) ซึ่งใช้ชื่อย่อของคำสั่งมาเขียนเรียงต่อกัน เมื่อเขียนเสร็จแล้วก็จะใช้โปรแกรมพิเศษ ซึ่งเรียกว่า แอสเซมเบลอร์ (Assembler) ทำการแปลคำสั่งจากภาษาแอสเซมบลี เป็นภาษาเครื่องโดยอัตโนมัติ จึงทำให้การเขียนโปรแกรมสะดวกขึ้นมาก แต่การเขียนโปรแกรมภาษาแอสเซมบลี ยังต้องเขียนแบบมีรายละเอียดทุกขั้นตอน หากมีคำสั่งให้ทำงานเดียวกันซ้ำกันหลายๆ ครั้ง ก็ต้องเขียนคำสั่งนั้นซ้ำหลายๆ ครั้ง นอกจากนั้น ผู้เขียนคำสั่งต้องมีความรู้เกี่ยวกับฮาร์ดแวร์เป็นอย่างดี เพราะจะต้องระบุว่าให้นำข้อมูลไปเก็บไว้ที่หน่วยความจำส่วนใด หรือนำข้อมูลมาจากหน่วยความจำส่วนใด

ภาษาเขียนโปรแกรมระดับสูง (High Level Language) เป็นภาษาที่พัฒนาขึ้นมาให้มีคุณสมบัติใกล้เคียงภาษามนุษย์มากขึ้น และเพิ่มสิ่งอำนวยความสะดวกต่างๆ เช่น ผู้ใช้ไม่จำเป็นต้องมีความรู้เกี่ยวกับฮาร์ดแวร์ โดยมอบภาระการจัดการใช้หน่วยความจำ ฯลฯ ให้กับระบบแปลภาษา การทำงานซ้ำก็สามารถสั่งได้ด้วยคำสั่งสั้นๆ เป็นต้น

ภาษาเขียนโปรแกรมระดับสูงยุคแรก ๆ ที่ได้รับความนิยมมาก ได้แก่ ALGOL FORTRAN และ COBOL ภาษาทั้งสามนี้ พัฒนาขึ้นมาสำหรับใช้กับคอมพิวเตอร์เครื่องใหญ่ เป็นภาษาที่มีขีดความสามารถสูง แต่มีกฎเกณฑ์ที่สลับซับซ้อนยากต่อการเรียนรู้สำหรับผู้เริ่มเรียน นอกจากนั้น ยังเปลืองหน่วยความจำมาก ไม่เหมาะที่จะใช้ไมโครคอมพิวเตอร์ซึ่งมีหน่วยความจำน้อย

1.5 ภาษาเบสิค (BASIC)

ในทศวรรษ 1960 มหาวิทยาลัย Dartmouth College ในสหรัฐอเมริกา ได้พัฒนาภาษาเขียนโปรแกรมขึ้นมาภาษาหนึ่งให้ชื่อว่า Beginner's All Purpose Symbolic Instruction Code (BASIC) โดยตั้งวัตถุประสงค์ว่า ให้เป็นภาษาเขียนโปรแกรมที่สามารถเรียนรู้ได้ง่ายสำหรับผู้เริ่มเรียน ผลปรากฏว่า ทำให้ผู้เริ่มเรียนสามารถเรียนรู้วิธีการเขียนโปรแกรมได้อย่างรวดเร็วจริง จึงกลายเป็นภาษาที่แพร่หลายในวงการศึกษา แต่ในระยะนั้นภาษาเบสิค ยังมีขีดความสามารถต่ำกว่าภาษาเขียนโปรแกรมอื่นๆ มาก จึงไม่เป็นที่ยอมรับในการใช้งาน

ต่อมาเมื่อประมาณ ปี ค.ศ. 1975 ไมโครคอมพิวเตอร์เริ่มออกสู่ตลาดในสหรัฐอเมริกา แต่เนื่องจากไมโครคอมพิวเตอร์มีหน่วยความจำน้อยมาก การเขียน

โปรแกรม จึงใช้ภาษาเครื่อง และภาษาแอสเซมบลี ผู้ผลิตเครื่องพยายามมองหาภาษาเขียนโปรแกรมระดับสูงที่จะนำมาใช้กับไมโครคอมพิวเตอร์ ในที่สุดพบว่า ภาษาเบสิคมีความเหมาะสมที่สุด เพราะนอกจากเรียนรู้ได้ง่ายแล้ว ยังใช้หน่วยความจำไม่มาก ภาษาเบสิคที่ประสบความสำเร็จในเชิงพาณิชย์อันดับแรกนั้น เป็นผลงานของ William H Gates III ซึ่งเป็นผู้ก่อตั้ง และเป็นเจ้าของบริษัท Microsoft ในปัจจุบัน

ภาษาเบสิคที่ใช้ในเครื่องไมโครคอมพิวเตอร์ส่วนใหญ่ มีลักษณะเป็น "ระบบแปลสด" (Interpreter) กล่าวคือจะแปลคำสั่งเป็นภาษาเครื่องทีละบรรทัด แล้วทำตามคำสั่งทันที แทนที่จะเป็น "ระบบแปลทั้งโปรแกรม"(Compiler) เหมือนภาษาระดับสูงส่วนมาก แต่มีภาษาเบสิคบางแบบ (เช่น CBASIC และ BASCOM) เป็น "ระบบแปลทั้งโปรแกรม" ซึ่งจะแปลคำสั่งทั้งหมดเป็นภาษาเครื่องเสียก่อน แล้วจึงจะทำตามคำสั่งในภาษาเครื่องนั้น

ภาษาเบสิค ที่ได้รับความนิยมสูงมากแบบหนึ่งคือภาษาเบสิคของแอปเปิลทู (Apple II) ซึ่งมีชื่อเฉพาะเรียกว่า แอปเปิลซอฟท์ (Apple Soft) แอปเปิลซอฟท์เป็นภาษาเบสิคที่มีขีดความสามารถค่อนข้างสูง และยังมีจุดเด่นเกี่ยวกับการแสดง ภาพ (Graphic) บนจออีกด้วย

1.6 คอมพิวเตอร์กับภาษาไทย

นับตั้งแต่มีไมโครคอมพิวเตอร์อุบัติขึ้นในโลก นักวิชาการของไทยก็พยายามค้นคว้าที่จะทำให้คอมพิวเตอร์สามารถใช้ภาษาไทยได้ ห้องปฏิบัติการไมโครคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ เป็นแห่งแรกที่ประสบความสำเร็จในการแสดงผลภาษาไทยบนเครื่องไมโครคอมพิวเตอร์ และต่อมาชมรมคอมพิวเตอร์โรงพยาบาลรามาธิบดี ก็สามารถสร้างโปรแกรม WORDRAMA ซึ่งเป็นโปรแกรม WORD PROCESSOR สำหรับเรียบเรียงและพิมพ์เอกสารภาษาไทยปนอังกฤษ โดยมีขีดความสามารถสูงมาก เทียบเท่า WORD PROCESSOR ภาษาอังกฤษชั้นเยี่ยมทีเดียว นับว่าคนไทยควรจะภาคภูมิใจที่มีผลงานคอมพิวเตอร์ภาษาไทย แต่ผลงานที่กล่าวมาแล้วทั้งหมดยังต้องใช้คำสั่งที่เป็นภาษาอังกฤษ เช่น ถ้าต้องการให้คอมพิวเตอร์พิมพ์ข้อความภาษาไทย ต้องใช้คำสั่งดังนี้

```
10 PRINT "คอมพิวเตอร์เครื่องนี้พิมพ์ภาษาไทยได้"
```

นักวิชาการของไทย ได้พยายามเปลี่ยนมาใช้คำสั่งภาษาไทย แต่ติดขัดที่วิธีการเก็บคำสั่งในภาษาเบสิค ซึ่งแทนคำสั่งด้วยรหัสเลข และรหัสเลขเหล่านี้จะซ้ำซ้อนกับรหัสเลขที่ใช้แทนอักษรไทยในเครื่องไมโครคอมพิวเตอร์ทั่วไป (ดูบทความเรื่อง คุณรู้จักเบสิคดีเพียงไร ตอนเบสิค VS ไทยเบสิค โดย ยืน ภู่วรวรรณ (ไมโครคอมพิวเตอร์ ฉบับที่ 2 กรกฎาคม 2526)

1.7 คอมพิวเตอร์คำสั่งไทย "ลำไย 28"

ในปี 2528 กลุ่มนักวิจัยแห่งมหาวิทยาลัยเชียงใหม่ ได้พบวิธีทำให้คอมพิวเตอร์ สามารถรับรู้และปฏิบัติตามคำสั่งภาษาไทยได้ จึงได้ออกแบบสร้างเครื่องไมโครคอมพิวเตอร์ขึ้น และให้ชื่อว่า "ลำไย 28"

"ลำไย 28" เป็นไมโครคอมพิวเตอร์ที่ได้รับการออกแบบทางฮาร์ดแวร์ให้มีคุณสมบัติ ดังต่อไปนี้

1. สามารถรับรู้และปฏิบัติตามคำสั่งที่เป็นภาษาไทย
2. สามารถแปลคำสั่งภาษาไทย ออกเป็นคำสั่งภาษาอังกฤษ
3. สามารถแปลคำสั่งภาษาอังกฤษออกเป็นคำสั่งภาษาไทย
4. สามารถแสดงข้อความบนจอทั้งภาษาไทย และภาษาอังกฤษในเวลาเดียวกัน

1.8 ภาษาเบสิคคำสั่งไทย "ไทยปกาสิต"

เมื่อมีเครื่องคอมพิวเตอร์ที่สามารถรับรู้ และปฏิบัติตามคำสั่งภาษาไทยได้แล้ว กลุ่มนักวิจัยผู้สร้าง "ลำไย 28" จึงได้สร้างภาษาเบสิคคำสั่งไทยขึ้น และให้ชื่อว่า "ไทยปกาสิต"

"ไทยปกาสิต" เป็นภาษาเขียนโปรแกรมที่ออกแบบขึ้นโดยมุ่งหวังให้คนไทยโดยเฉพาะอย่างยิ่งเยาวชนไทย สามารถเรียนรู้เกี่ยวกับการเขียนโปรแกรมคอมพิวเตอร์อย่างง่ายดาย โดยปราศจากอุปสรรคที่เกิดจากความไม่สันทัดในภาษาต่างประเทศ ด้วยวัตถุประสงค์ดังกล่าวจึงได้เลือกเอาคำง่าย ๆ ที่สามารถสื่อความหมายได้ชัดเจนมาบัญญัติขึ้นเป็นคำสั่ง

สำหรับผู้ที่คุ้นเคยกับภาษาเบสิค ที่ใช้คำสั่งภาษาอังกฤษอยู่แล้ว อาจรู้สึกหงุดหงิดอยู่บ้าง ที่จะต้องเรียนรู้คำภาษาไทยที่จะมาใช้เป็นคำสั่ง แต่หากท่านจะนึกถึงตอนที่ท่านเรียนภาษาเบสิคใหม่ ๆ กว่าท่านจะคุ้นกับคำต่าง ๆ เช่น GET,GOSUB,

TAB ฯลฯ ก็ไม่ใช่ว่าจะรู้โดยธรรมชาติ หากแต่ต้องอาศัยความพยายามในการเรียนรู้ไม่น้อยเหมือนกัน ฉะนั้น สำหรับเยาวชน หรือผู้เริ่มเรียนใหม่แล้ว ความรู้สึกอคติเช่นนี้จะไม่มีเลย

ในขณะเดียวกัน ผู้สร้าง "ลำไย 28" และ "ไทยปกาสิต" ได้ตระหนักดีว่า คนไทยควรมีโอกาสได้ศึกษาวิธีการเขียนโปรแกรมของชาวต่างประเทศ เพื่อเรียนรู้เทคนิคต่างๆ จึงได้กำหนดให้ "ลำไย 28" มีความสามารถในการแปลคำสั่งจากไทยเป็นอังกฤษ และอังกฤษเป็นไทย เพื่อให้ผู้เรียนรู้ "ไทยปกาสิต" ได้มีโอกาสเห็นโลกของการเขียนโปรแกรมภาษาเบสิคอันหาขอบเขตมิได้ และเพื่อให้เขาเหล่านั้นได้มีโอกาสเพิ่มพูนประสบการณ์ด้วยวิธีการดังกล่าว

นี่เป็นอีกครั้งหนึ่ง ที่คนไทยควรจะภาคภูมิใจอย่างยิ่ง เพราะนับว่าเป็นครั้งแรกในโลก ที่มีการสร้างภาษาเขียนโปรแกรม ด้วยคำสั่งที่ไม่ใช่ภาษาอังกฤษ (ทั้ง ๆ ที่ชาวฝรั่งเศสขึ้นชื่อว่าชาตินิยมจัดในเรื่องภาษา แต่เวลาเขียนโปรแกรมยังต้องยอมใช้คำสั่งภาษาอังกฤษ !)

1.9 ของคนไทย โดยคนไทย และเพื่อคนไทย

ณ บัดนี้ วิวัฒนาการด้านคอมพิวเตอร์ของไทยได้ดำเนินมาถึงจุดที่เรามีระบบ คอมพิวเตอร์ทั้งระบบ (คือทั้งตัวเครื่องและภาษาเขียนโปรแกรม) ที่เป็นไทยโดยสมบูรณ์แล้ว เราสามารถจะกล่าวได้เต็มปากว่าระบบนี้เป็นของคนไทย โดยคนไทย และเพื่อคนไทย อีกทั้งเป็นอนาคตของประเทศไทยอีกด้วย แล้วท่านในฐานะที่เป็นคนไทยจะยินดีช่วยสนับสนุนไหม ?

บทที่ 2
วิธีใช้เครื่องคอมพิวเตอร์ "ลำไย 28"

2.1 คุณสมบัติพิเศษของ "ลำไย 28"

"ลำไย 28" เป็นเสมือนไมโครคอมพิวเตอร์ 3 เครื่อง ในเครื่องเดียวกัน กล่าวคือสามารถทำงานด้วยระบบแปลภาษาที่แตกต่างกันได้ถึง 3 ระบบ คือ

ระบบที่ 1 เป็นคอมพิวเตอร์ภาษาอังกฤษล้วน (เหมือนกับเครื่องคอมพิวเตอร์ที่มาจากต่างประเทศ

ระบบที่ 2 เป็นคอมพิวเตอร์คำสั่งอังกฤษ - ข้อความไทย/อังกฤษ (เหมือนกับคอมพิวเตอร์ส่วนใหญ่ในประเทศไทย ที่ได้รับการดัดแปลงให้สามารถพิมพ์ข้อความไทย/อังกฤษได้)

ระบบที่ 3 เป็นคอมพิวเตอร์**คำสั่งไทย**-ข้อความไทย/อังกฤษ ระบบนี้เป็นเอกลักษณ์ ของ "ลำไย 28" ซึ่งพิเศษกว่าเครื่องอื่นๆ ทุกเครื่องที่มีอยู่ในปัจจุบัน

2.2 แป้นพิมพ์

แป้นพิมพ์ของ "ลำไย 28" จัดวางตัวอักษรไทยและอังกฤษ เหมือนกับแป้นพิมพ์ของเครื่องพิมพ์ดีดไฟฟ้าโอลิมเปีย ฉะนั้นผู้ที่สามารถพิมพ์สัมผัสภาษาไทย/อังกฤษ บนเครื่องพิมพ์ดีดไฟฟ้าได้จะสามารถใช้แป้นพิมพ์ของ "ลำไย 28" ได้อย่างคล่องแคล่ว

แป้นพิมพ์ที่ใช้เป็นแป้นพิมพ์รุ่นใหม่ ซึ่งมีแป้นพิเศษเป็นจำนวนมาก รวมทั้งบางแป้นที่ผู้ใช้สามารถกำหนดตัวอักษรหรือข้อความได้เอง ในที่นี้จะอธิบายเฉพาะแป้นพิเศษที่จำเป็นสำหรับการใช้งานกับ "ไทยปกาสิต" เท่านั้น คือ

CTRL เรียกว่า แป้น **คอนโทรล** (CONTROL KEY) ต้องใช้ร่วมกับแป้นอื่นโดยกดแป้นนี้แช่ไว้ก่อน แล้วจึงกดแป้นอื่น

ESC เรียกว่า แป้น **เอสเคป** (ESCAPE KEY) การใช้ต่างกับแป้นคอนโทรล โดยกดแป้นนี้ ปล่อยออกก่อน แล้วจึงกดแป้นอื่นต่อไป

RETURN เรียกว่าแป้น **รีเทอน** (RETURN KEY) เมื่อกดแป้นนี้จะเป็นการบอกเครื่องว่า บรรทัดที่กำลังพิมพ์อยู่จบลงแล้ว เครื่องก็จะนำข้อความบรรทัดนั้นไปตรวจสอบ หากพบว่าเป็นคำสั่ง หรือส่วนต่อเนื่องของคำสั่ง ก็จะ

ปฏิบัติตามคำสั่งนั้นทันที

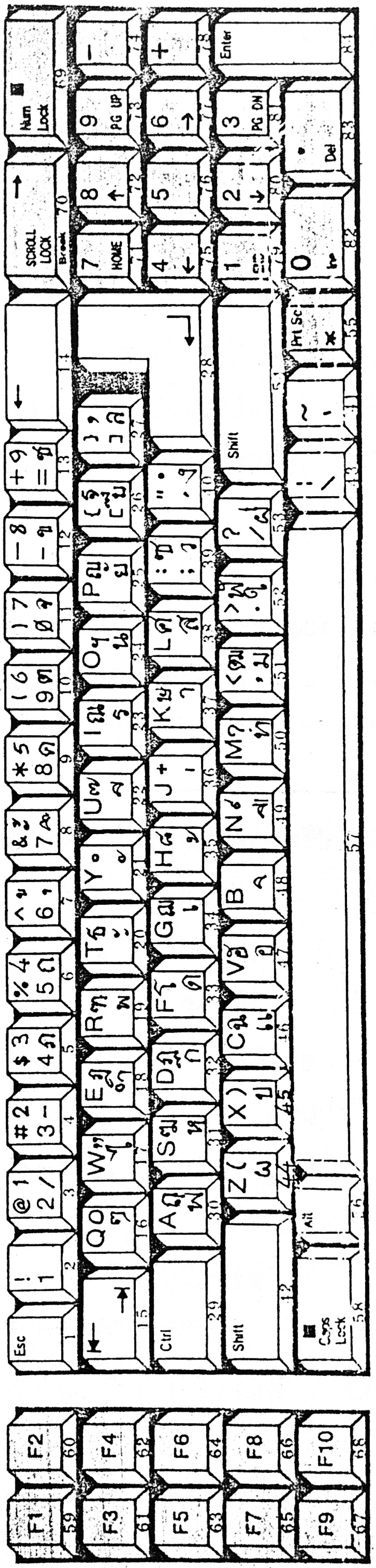
SHIFT เรียกว่าแป้น **ชิพท์** (SHIFT KEY) ทำหน้าที่เช่นเดียวกับแป้นยกแคร่ ในพิมพ์ดีด

<-- เรียกว่าแป้น **ลูกศรซ้าย** (LEFT ARROW KEY) ใช้ถอยหลังกลับซึ่งจะลบอักษร หรือเครื่องหมายที่ได้พิมพ์ไปแล้วออกจากหน่วยความจำ

--> เรียกว่าแป้น **ลูกศรขวา** (RIGHT ARROW KEY) ใช้เลื่อนตำแหน่งที่จะพิมพ์ไปข้างหน้า ถ้าเลื่อนผ่านช่องว่างจะมีผลเท่ากับการเคาะคาน **เว้นวรรค** (SPACE BAR) แต่ถ้าเลื่อนผ่านตัวอักษรตัวใดบนจอจะมีผลเท่ากับได้พิมพ์ตัวอักษรนั้นลงไป เครื่องจะรับอักษรนั้นเข้าหน่วยความจำ

NUM LOCK เป็นแป้น **เรียกใช้** หรือ **เลิกใช้ แป้นตัวเลข** ที่อยู่ทางด้านขวาของแป้นตัวอักษร เมื่อกด 1 ที ไฟแดงบนแป้น **NUM LOCK** จะติด แสดงว่า แป้นตัวเลขพร้อมที่จะให้ตัวเลขออกมาเมื่อมีการกด เมื่อกดซ้ำอีก 1 ที ไฟแดงดับ แสดงว่าเลิกใช้แป้นตัวเลข และเปลี่ยนหน้าที่ของแป้นตัวเลขไปใช้อย่างอื่น เช่น เปลี่ยนไปเป็นลูกศรเลื่อนตำแหน่งพิมพ์ 4 ทิศ เป็นต้น ลูกศรเหล่านี้ จะใช้ได้ภายใต้การควบคุมของโปรแกรมสำเร็จรูปบางโปรแกรมเท่านั้น

CAPLOCK เป็นแป้นสำหรับล็อคให้พิมพ์ตัวอักษรภาษาอังกฤษตัวใหญ่ได้โดยไม่ต้อง กด **SHIFT** เวลาใช้แป้นนี้ไฟแดงจะติด


Esc
Ctrl
Shift
Alt
Caps Lock
Enter
Num Lock
SCROLL LOCK
Break
PG UP
PG DN
HOME
Del
Prt Sc
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10

รูป 2.1 แป้นพิมพ์

2.3 การเปลี่ยนภาษาของแป้นพิมพ์

การเปลี่ยนภาษาของแป้นพิมพ์ จากภาษาอังกฤษเป็นภาษาไทย หรือไทยเป็นอังกฤษ ทำได้โดยการกด (CTRL-A) เครื่องหมายวงเล็บเน้นให้ทราบว่าต้องกด CTRL แช่ไว้ แล้วกด A ตาม เสร็จแล้วจึงปล่อยทั้งคู่ การเปลี่ยนภาษานี้เรียกว่า เปลี่ยน**โหมด** (MODE) ของแป้นพิมพ์

ในขณะที่คอมพิวเตอร์พร้อมที่จะพิมพ์ตัวอักษรบนจอ จะมีเครื่องหมายบอกตำแหน่งที่พิมพ์เรียกว่า เคอร์เซอร์ (CURSOR) เมื่ออยู่ในโหมดภาษาอังกฤษ เคอร์เซอร์จะมีรูปร่าง รูปสี่เหลี่ยมกระพริบ ดังนี้ ▭ แต่เมื่ออยู่ในโหมดภาษาไทยจะมีเส้นประใต้รูปสี่เหลี่ยมกระพริบ ดังนี้ ▭

ขณะเปิดเครื่องใหม่ แป้นพิมพ์จะอยู่ในโหมดภาษาอังกฤษ และทุกครั้งที่มีการกดแป้น**รีเทอน** แป้นพิมพ์จะกลับคืนสู่โหมดภาษาอังกฤษ

2.4 การเลือกระบบแปลภาษา

ในข้อ 2.1 ได้กล่าวแล้วว่า "ลำไย 28" มีระบบแปลภาษาให้เลือกได้ถึง 3 ระบบ เมื่อเปิดเครื่องครั้งแรกเครื่องจะเข้าสู่ระบบที่ 3 โดยอัตโนมัติ และบน จอจะปรากฏข้อความดังนี้

สำนักที่ปรึกษาคอมพิวเตอร์เชียงใหม่
เสนอ
ไทยปกาสิต
โดย
ขจรศักดิ์ คันธพนิต
ถวัลย์วงศ์ ไกรโรจนานันท์
ชาญชาย ณ บางช้าง
ชาญศักดิ์ สุวรรณกุล
ฉบับที่ 1 สงวนลิขสิทธิ์ พ.ศ. 2528
⊳

เครื่องหมาย ⊳ ที่นำหน้า**เคอร์เซอร์** มีชื่อเรียกว่า **พร้อมท์** (PROMPT) เป็นเครื่องหมายที่แสดงว่า "ไทยปกาสิต" กำลังรอรับคำสั่ง

หากต้องการเปลี่ยนระบบแปลภาษา ให้กดแป้น ดังนี้

ESC (CTRL E) เข้าสู่ระบบที่ 1 (คำสั่งอังกฤษ ข้อความอังกฤษ)

ESC (CTRL R) เข้าสู่ระบบที่ 2 (คำสั่งอังกฤษ ข้อความไทย/อังกฤษ)

หากต้องการกลับคืนสู่ระบบที่ 3 ก็ให้กด

ESC (CTRL T) เข้าสู่ระบบที่ 3 (คำสั่งไทย ข้อความไทย/อังกฤษ)

ให้สังเกตว่า **พร้อมท์** ของระบบ ทั้ง 3 ต่างกันดังนี้

ระบบที่ 1] อังกฤษล้วน

ระบบที่ 2 [คำสั่งอังกฤษ ข้อความไทย - อังกฤษ

ระบบที่ 3 ▷ **ไทยปกาสิต** คำสั่งไทย ข้อความไทย-อังกฤษ

การเปลี่ยนระบบจะทำได้เฉพาะในขณะที่บรรทัดสุดท้ายมี **พร้อมท์** ตัวใดตัวหนึ่ง อยู่ที่ต้นบรรทัด และตามด้วย **เคอร์เซอร์** เท่านั้น หากมีข้อความอื่นใดให้กด (CTRL - BREAK) ซึ่งเป็นการออกจากโปรแกรม (RESET) เสียก่อนแล้วจึงค่อยทำการเปลี่ยนระบบดังกล่าวข้างต้น

2.5 การแสดงตัวอักษรบนจอภาพ

ในระบบที่ 1 (ภาษาอังกฤษล้วน) แต่ละบรรทัดมี 40 ตัวอักษร เต็มจอจะมี 24 บรรทัด

ในระบบที่ 2 ระบบที่ 3 (ภาษาอังกฤษ/ไทย) แต่ละบรรทัดมี 40 ตัวอักษรเต็มจอจะมีได้สูงสุด 12 บรรทัด แต่ถ้าบรรทัดใดมีสระล่าง (ุ , ู) จำนวนบรรทัดจะลดลง โดยอย่างน้อยที่สุดจะมี 8 บรรทัด

เมื่อข้อมูลปรากฏเต็มจอแล้ว จะเลื่อนขึ้นเองโดยอัตโนมัติ

2.6 การแปลคำสั่งจาก อังกฤษเป็นไทย หรือจากไทยเป็นอังกฤษ

โปรแกรมที่เขียนด้วย"ไทยปกาสิต" สามารถแปลเป็นคำสั่งภาษาอังกฤษได้ เมื่อเปลี่ยนระบบเข้าสู่ระบบที่ 2 โดยกด ESC (CTRL - R) แล้วพิมพ์คำสั่ง LIST เข้าไป ในทำนองเดียวกัน โปรแกรมที่เขียนด้วย ภาษาเบสิคคำสั่งอังกฤษก็สามารถแปลเป็นคำสั่งไทยได้โดยการกด ESC(CTRL - T) แล้วพิมพ์ **แจง** เข้าไป

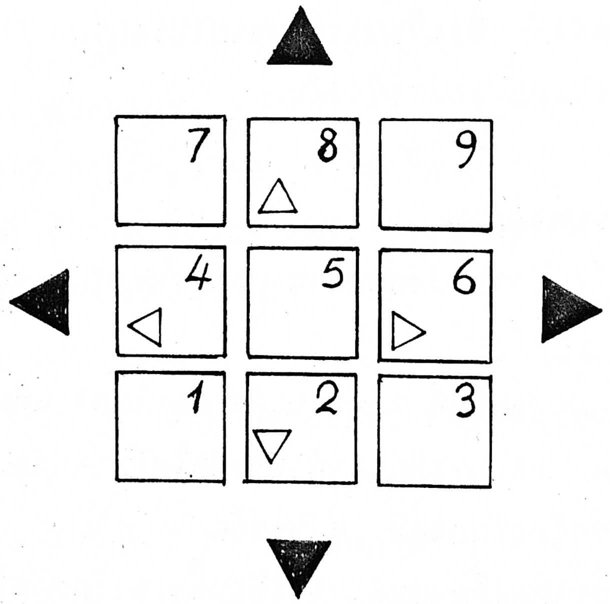
ในการแปลคำสั่งดังกล่าวจะแปล เฉพาะตัวคำสั่งเท่านั้น ส่วนข้อความ

ต่างๆ จะอยู่ในภาษาเดิม

"ไทยปกาสิต" ใช้ชื่อตัวแปรเป็นอักษรภาษาอังกฤษ และใช้คำเดิมแทนฟังก์ชั่นคณิตศาสตร์ต่าง ๆ เช่น SIN,COS,TAN,INT,EXP เป็นต้น โดยไม่มีการแปล ทั้งนี้เพราะเห็นว่า แม้แต่ในตำราคณิตศาสตร์ภาษาไทยก็ทำเช่นนี้

2.7 การเลื่อนเคอร์เซอร์

การเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการบนจอ ให้กดแป้นลูกศร 4 ทิศทางที่อยู่ในตำแหน่งเดียวกับแป้นตัวเลข 2,4,6,8 ก่อนใช้แป้นเหล่านี้ให้กดปุ่ม NUMLOCK เสียก่อน (ดูรูป 2.2)



รูป 2.2 การเลื่อนเคอร์เซอร์

บทที่ 3
คำแนะนำเบื้องต้นในการใช้
"ไทยปกาสิต"

3.1 สำหรับผู้ที่รู้ภาษาเบสิคแล้ว

เนื่องจากโครงสร้างของ "ไทยปกาสิต" เหมือนกับโครงสร้างของภาษาเบสิค ผู้ที่รู้ภาษาเบสิคอยู่แล้ว จะสามารถใช้ "ไทยปกาสิต"ได้ทันที โดยเพียงแต่เทียบคำสั่งภาษาไทยกับคำสั่งภาษาอังกฤษ ในตาราง 3.1

การบัญญัติศัพท์ต่าง ๆ ขึ้นมาใช้เป็นคำสั่งในภาษา "ไทยปกาสิต" นั้นได้อาศัยหลัก 2 ประการ คือ

1. คำศัพท์ทุกคำ จะต้องสื่อความหมายได้อย่างชัดเจน เพื่อให้ผู้ใช้สามารถเข้าใจผลที่เกิดขึ้นจากการปฏิบัติตามคำสั่งนั้น

2. การพิมพ์คำสั่งภาษาไทย บนแป้นพิมพ์ควรจะทำได้สะดวก โดยมีการเปลี่ยนโหมดของแป้นพิมพ์น้อยที่สุด

ด้วยเหตุนี้ จึงคาดหวังได้ว่า ผู้เรียน "ไทยปกาสิต" จะสามารถเรียนรู้ได้อย่างรวดเร็วกว่าการเรียนภาษาเบสิคที่ใช้คำสั่งภาษาอังกฤษ

3.2 คำแนะนำเบื้องต้นสำหรับผู้เริ่มเรียน

การพิมพ์คำสั่งคอมพิวเตอร์นั้น จะต้องพิมพ์ให้ถูกต้องจริงๆ เพราะคอมพิวเตอร์ไม่มี**ดุลยพินิจ** สมมติว่าเราพิมพ์คำว่า "พิม พ์" ซึ่งมีเว้นวรรคแทรกอยู่ระหว่าง ม กับ พ แม้ว่าเราจะอ่านออก แต่คอมพิวเตอร์ส่วนใหญ่ จะถือว่าเป็นคนละคำกับ "พิมพ์" แม้ว่าคอมพิวเตอร์เพื่อการศึกษาบางเครื่อง (รวมทั้ง "ลำไย 28") จะมีคุณสมบัติพิเศษ คือ ไม่คำนึงถึงการเว้นวรรคภายในคำสั่ง และระหว่างคำสั่ง แต่การหัดพิมพ์ให้ถูกต้องจะเป็นการเพาะนิสัยที่ดี ซึ่งจะทำให้สามารถใช้คอมพิวเตอร์ทุกเครื่องได้โดยไม่มีปัญหา

3.3 คำสั่งตรงกับคำสั่งในโปรแกรม

คำสั่งตรง เป็นคำสั่งประเภทที่สั่งแล้ว เครื่องจะทำงานทันที เช่น ลองพิมพ์ประโยคต่อไปนี้

⇨ พิมพ์ "คอมพิวเตอร์"

เมื่อจบประโยคแล้ว กด RETURN ภาพบนจอจะเป็นดังนี้

⊳ พิมพ์ "คอมพิวเตอร์"
คอมพิวเตอร์
⊳ ▭

แสดงว่าคอมพิวเตอร์ได้ปฏิบัติตามคำสั่งแล้ว โดยการพิมพ์คำว่า **"คอมพิวเตอร์"** ในบรรทัดถัดมา

คำสั่งตรงที่ควรจะเรียนรู้เพิ่มเติมในขั้นตอนนี้ คือ

ใหม่ ให้ยกเลิกโปรแกรมที่อยู่ในหน่วยความจำเพื่อเตรียมรับโปรแกรมใหม่

แจง ให้พิมพ์คำสั่งต่างๆ ในโปรแกรมที่อยู่ในหน่วยความจำเรียงตามลำดับเลข บรรทัด

ทำ ให้ปฏิบัติตามคำสั่งในโปรแกรมที่อยู่ในหน่วยความจำ

คำสั่งทั้งสามข้างบนนี้ เป็นคำสั่งตรง แต่เป็นคำสั่งที่เกี่ยวกับโปรแกรม จึงจะขออธิบาย ความหมายของ "คำสั่งในโปรแกรม" ดังนี้

คำสั่งในโปรแกรม คือ คำสั่งที่เราเขียนเรียบเรียงไว้เป็นชุด ให้ไปเก็บไว้ในหน่วยความจำ คอมพิวเตอร์จะยังไม่ทำตามคำสั่งเหล่านี้จนกว่า เราจะสั่งว่า **ทำ** การทำเช่นนี้ เป็นการอำนวยความสะดวก ทำให้เราสามารถสั่งงานไว้ล่วงหน้าทีละมากๆ โดยการพิมพ์คำสั่งติดต่อกันไปเก็บไว้ในหน่วยความจำ ขบวนการนี้เรียกว่า "การพิมพ์ โปรแกรม" ซึ่งอาจจะเริ่มจากการเขียนโปรแกรมลงบนแผ่นกระดาษก่อน แล้วจึงมาพิมพ์เข้าเครื่องก็ได้ กฎการเขียนโปรแกรม ในภาษา "ไทยปกาสิต" ก็เช่นเดียวกับภาษาเบสิคอื่นๆ กำหนดให้เลขบรรทัดเป็นตัวบอกลำดับของคำสั่ง ฉะนั้นโปรแกรมภาษาไทยปกาสิตโปรแกรมแรกของเรา จึงอาจมีหน้าตาบนจอดังนี้

⊳ ใหม่
10 พิมพ์ "สวัสดีครับ"
20 พิมพ์ "ผมชื่อ";
30 พิมพ์ "(ชื่อผู้เขียนโปรแกรม)"
▭

จงสังเกตว่า ข้อความที่จะสั่งให้พิมพ์นั้น ต้องอยู่ระหว่างเครื่องหมายอัญประกาศ "....." คราวนี้ลองใช้คำสั่ง **ทำ** ดู จะได้ผลดังนี้

□ ทำ

สวัสดีครับ

ผมชื่อ (ชื่อผู้เขียนโปรแกรม)

3.4 เลขบรรทัดกับเครื่องวรรคตอน

โปรแกรมภาษา "ไทยปกาสิต" ประกอบด้วยบรรทัดของคำสั่ง ที่เริ่มต้นด้วยหมายเลขบรรทัด ตามด้วยตัวคำสั่ง และจบลงด้วยการกด **RETURN** เลขบรรทัดที่สามารถใช้ได้คือ เลขจำนวนเต็มตั้งแต่ 1 จนถึง 65536 แต่ตามปกตินิยมใช้เลขบรรทัดเว้นทีละ 10 ทั้งนี้เพื่อเหลือเลขบรรทัดไว้แทรกในการปรับปรุงแก้ไขโปรแกรม เวลาพิมพ์โปรแกรม เราจะไม่พิมพ์เรียงลำดับเลขบรรทัดก็ได้ แต่โปรแกรมจะทำงานเรียงตามลำดับเลขบรรทัดเสมอ

ในแต่ละบรรทัดอาจมีคำสั่งหลายคำสั่งก็ได้ แต่ต้องใช้เครื่องหมาย : (เรียกว่า โคลอน หรือ COLON) คั่นระหว่างคำสั่ง เช่น

40 พิมพ์ "ลาก่อน" : พิมพ์ "สวัสดี"

คำสั่งนี้มีผลเช่นเดียวกับ คำสั่ง 2 บรรทัด ต่อไปนี้

40 พิมพ์ "ลาก่อน"
50 พิมพ์ "สวัสดี

นอกจากเครื่องหมาย : ซึ่งใช้คั่นระหว่างคำสั่งในบรรทัดเดียวกันแล้ว ยังมีเครื่องหมายวรรคตอนอีกสองตัวคือ ; (เซมิโคลอน หรือ SEMICOLON) กับ , (คอมมา หรือ COMMA)

ขอให้ย้อนกลับไปดูโปรแกรมที่ท้าย ข้อ 3.3 ที่ท้ายบรรทัด 20 มีเครื่องหมาย ; ซึ่งแสดงว่า **เมื่อจบข้อความแล้ว ไม่ต้องขึ้นบรรทัดใหม่ ไม่ต้องเว้นวรรค** ดังนั้น ข้อความที่บรรทัด 30 สั่งให้พิมพ์ คือ (ชื่อผู้เขียนโปรแกรม) จึงพิมพ์ติดต่อกับคำว่า "ผมชื่อ" ของบรรทัด 20 เราอาจเขียนบรรทัด 20 และ 30 รวม

เป็นบรรทัดเดียวกัน โดยใช้คำสั่ง **พิมพ์** ครั้งเดียวก็ได้ดังนี้

20 พิมพ์ " ผมชื่อ " ; " (ชื่อผู้เขียนโปรแกรม) "

บรรทัด 30 ซึ่งจะเลิกใช้ สามารถลบออกได้ โดยพิมพ์

30 RETURN

เมื่อลองสั่ง **ทำ** ดูใหม่จะพบว่าได้ผลเหมือนเดิม แสดงว่า คำสั่ง**พิมพ์** คำสั่งเดียวสามารถสั่งให้พิมพ์ข้อความได้หลาย ๆ ข้อความโดยมี **;** คั่นระหว่างข้อความ เราอาจเขียนบรรทัด 40 ใหม่ด้วยวิธีนี้ (ให้ทดลองเขียนดู และสั่ง**ทำ** ดู)

เครื่องหมาย **,** ใช้คั่นระหว่างข้อความที่สั่งพิมพ์เช่นเดียวกัน แต่มีความหมายว่า ให้แบ่งความกว้างของจอเป็น 3 คอลัมน์ คอลัมน์แรกเริ่มจากซ้ายสุดของจอมีความกว้าง 16 ตัวอักษร คอลัมน์ที่สองเริ่มจากตำแหน่งที่ 17 มีความกว้าง 16 ตัวอักษร คอลัมน์ที่สามเริ่มจากตำแหน่งที่ 33 มีความกว่ง 8 ตัวอักษร จะมีการใช้คอลัมน์ที่สองก็ต่อเมื่อตำแหน่งที่ 16 เป็นช่องว่างและจะมีการใช้คอลัมน์ที่สามก็ต่อเมื่อตำแหน่งที่ 24-32 เป็นช่องว่าง

ทั้งเครื่องหมาย **;** และ **,** ใช้คั่นระหว่างข้อมูลที่เป็นตัวเลขได้เช่นเดียวกับข้อความ

จงทดลองพิมพ์โปรแกรมต่อไปนี้ แล้วสั่ง **ทำ** ดู

```
▷ ใหม่
100  พิมพ์  2528, 2529, 2530
110  พิมพ์  2528, 2529, 2530
200  พิมพ์ "ไทยปกาสิต", "ไทยปกาสิต", "ไทยปกาสิต"
210  พิมพ์ "ไทยปกาสิต", "ไทยปกาสิต", "ไทยปกาสิต"
□
```

หมายเหตุ จงสังเกตว่า ตัวเลขในบรรทัด 100 และ 110 ไม่มี "...."

เหตุผลจะได้อธิบายในข้อต่อไป

3.5 การคำนวณและผลลัพธ์

คอมพิวเตอร์สามารถทำการคำนวณได้ตั้งแต่ การ บวก ลบ คูณ หาร จนถึงสูตรคณิตศาสตร์ที่สลับซับซ้อน ในภาษาไทยปกาสิต เราสามารถสั่งให้คอมพิวเตอร์ ทำการคำนวณและพิมพ์ผลลัพธ์ได้โดยใช้คำสั่ง **พิมพ์** เช่น จงทดลองทำตามตัวอย่างต่อไปนี้

พิมพ์ 4+6 (บวก)

10

พิมพ์ 500-437 (ลบ)

63

พิมพ์ 100*23 (คูณ)

2300

พิมพ์ 96/12 (หาร)

8

พิมพ์ 3^2 (ยกกำลัง)

9

พิมพ์ 3*4*10-800 (ระคน)

-680

ตัวเลขที่ตามหลังคำสั่ง **พิมพ์** ถ้าไม่มี "...." ถือว่าเป็น**จำนวนเลข** ที่สามารถนำไปคำนวณได้ แต่ถ้ามี "...." ถือว่าเป็นข้อความ จะพิมพ์ตามนั้นโดยไม่มีการคำนวณ เช่น

พิมพ์ "4+6"

4+6

พิมพ์ 46 ; "46"

4646

จงอธิบายผลลัพธ์ บรรทัดสุดท้ายนี้ ถ้าอธิบายไม่ได้ ขอให้ย้อนกลับไปอ่านหัวข้อ 3.4 และ 3.5 ใหม่

ไทยปกาสิตยอมให้ใช้จำนวนเลข 9 หลัก (นับรวมกันทั้งก่อนและหลังจุดทศนิยม) จำนวนเลขที่เกิน 9 หลัก จะถูกปัดเศษให้เหลือไม่เกิน 9 หลัก เช่น

พิมพ์ 12.34567890

12.345679 (ปัดขึ้น, ทิ้งเลข 0 ข้างท้าย)

พิมพ์ 12.34567894

12.3456789 (ปัดลง)

จำนวนเลขที่มีค่ามากกว่า 123456789 จะถูกแสดงด้วยเลขยกกำลัง (แบบวิทยาศาสตร์) เช่น

พิมพ์ 123456789123

1.23456789E + 11

3.6 สัญญาณบอกข้อผิดพลาด

เมื่อ "ไทยปกาสิต" ตรวจพบข้อผิดพลาดในคำสั่ง จะมีเสียงดัง "ตู๊ด" เตือนขึ้น 1 ครั้ง พร้อมกับมีข้อความบอกความผิดพลาด ที่พบบ่อยที่สุดคือ การพิมพ์คำสั่งผิด เช่น

พิพม์ "ไทยปกาสิต" (คำสั่งว่า"พิมพ์"สะกดผิด)

? ไวยากรณ์ผิด

3.7 การแจง

การ**แจง** คือ การนำคำสั่งที่อยู่ในหน่วยความจำออกมาแสดงบนจอ โดยเรียงตามลำดับเลขบรรทัด ขณะนี้โปรแกรมที่อยู่ในหน่วยความจำ คือ คำสั่งตั้งแต่บรรทัด 100 ถึง 210 ทดลอง **แจง** ดู

เราอาจสั่ง **แจง** เพียงบรรทัดใดบรรทัดหนึ่งก็ได้ เช่น

แจง 200

หรือ แจง ระหว่างบรรทัดใดถึงบรรทัดใดก็ได้ เช่น

```
แจง 110, 200
```

ถ้าเลขหน้าเครื่องหมาย , ไม่มีถือว่าแจงตั้งแต่บรรทัดแรก และถ้าเลขหลังเครื่องหมาย , ไม่มีถือว่าแจงจนถึงบรรทัดสุดท้าย เช่น

```
แจง, 200
แจง 110,
```

ในระหว่างที่คำสั่งกำลังถูกนำขึ้นแสดงบนจอตามลำดับนั้น เราสามารถหยุดการเลื่อนขึ้นไว้ชั่วคราวได้โดยการกด (CTRL-S) และให้เลื่อนต่อไปได้โดยการกด**คานเว้นวรรค** วิธีนี้ช่วยให้เราสามารถดูโปรแกรมยาวๆ ได้โดยส่วนที่เราต้องการดูไม่เลื่อนขึ้นหายไปจากจอ

3.8 การแก้ไขโปรแกรม

ในการพิมพ์คำสั่งแต่ละบรรทัด หากพบว่าพิมพ์ผิด ก่อนที่จะกด **RETURN** ยังสามารถแก้ไขได้ โดยการใช้ <-- ถอยกลับไปถึงตำแหน่งของตัวอักษรที่ผิด พิมพ์ตัวอักษรที่ถูกทับลงไปในตำแหน่งนั้น แล้วใช้ --> เลื่อนผ่านตัวอักษรที่พิมพ์ถูกต้องแล้วทุกตัวที่อยู่ทางด้านขวาของตัวที่ถูกแก้นั้นไปจนสุดบรรทัดแล้วจึงกด RETURN

สำหรับบรรทัดคำสั่งที่พบคำผิดหลังจากกด **RETURN** อาจพิมพ์ใหม่หมดทั้งบรรทัด (รวมทั้งเลขบรรทัดด้วย) โดยวิธีนี้ บรรทัดที่พิมพ์ใหม่จะเข้าไปแทนบรรทัดเก่าที่ใช้เลขบรรทัดเดียวกัน

หากต้องการยกเลิกคำสั่งบรรทัดใด ให้พิมพ์เลขบรรทัดนั้นเปล่าๆ แล้วกด **RETURN**

นอกจากวิธีแก้ไขโดยการพิมพ์ใหม่ทั้งบรรทัดแล้ว เรายังสามารถทำการแทรกตัวอักษร หรือคำ ลงไปในคำสั่งที่พิมพ์เสร็จแล้วได้ ดังตัวอย่างต่อไปนี้

```
▷ ใหม่
10  พิมพ์ "ปกาสิต"
```

หากต้องการแก้ไข "ปกาสิต" เป็น "ไทยปกาสิต" ให้ทำดังนี้

1. ใช้วิธีเลื่อนเคอร์เซอร์ด้วยแป้นลูกศรสี่ทิศทางตามที่อธิบายในข้อ 2.7 เลื่อนให้เคอร์เซอร์ไปตรงกับเลข 1 ของเลขบรรทัด
2. ใช้แป้น --> ทำให้เคอร์เซอร์เดินผ่านคำสั่งบรรทัดนี้จนถึงตัว **ป** ของ **"ปกาสิต**
3. กดแป้นลูกศรชี้ขึ้นเพื่อเลื่อนเคอร์เซอร์ขึ้นบน 1 บรรทัด พิมพ์คำว่า **ไทย**
4. ใช้แป้นลูกศรลงกับแป้นลูกศรซ้าย(ในชุดสี่ทิศทาง) เลื่อนเคอร์เซอร์ลงมาตรงกับ **ป** ของปกาสิต
5. ใช้ --> เลื่อนเคอร์เซอร์ผ่านไปจนจบบรรทัดแล้วกด **RETURN**

จงทดลองทำตามตัวอย่างข้างต้นแล้ว **แจง** ดูใหม่

3.9 จำนวนเลข

จำนวนเลขที่สามารถเก็บไว้ในหน่วยความจำของคอมพิวเตอร์มี 2 ชนิด คือ **จำนวนเต็ม** กับ **จำนวนจริง**

จำนวนเต็ม หมายถึง จำนวนที่ไม่มีเศษส่วน หรือไม่มีตัวเลขตามหลังจุดทศนิยม "ไทยปกาสิต" อนุญาตให้ใช้เลขจำนวนเต็มได้ ระหว่าง -32767 กับ 32767 การพิมพ์เลขบนคอมพิวเตอร์ ห้ามพิมพ์ , คั่นระหว่างหลัก ตัวอย่างจำนวนเต็มได้แก่ 0, 1, 44, 32699, -15 เป็นต้น

จำนวนจริง คือ จำนวนเลขที่สามารถมีเศษส่วน หรือมีตัวเลขตามหลังจุดทศนิยมได้ จำนวนจริงที่มีค่าน้อยที่สุด และมากที่สุดที่ไทยปกาสิตยอมรับได้ คือ

-100000000000000000000000000000000000000

และ 100000000000000000000000000000000000000

(ตัวเลขข้างบนคือ 1 ตามด้วยเลข '0' 38 ตัว)

ตัวอย่างเลขจำนวนจริงได้แก่ 5, -15, 65000, 161, 0, 0.5, 0.165432, -0.000009, 1.6, 24.0055, -64.2 **เป็นต้น** ถ้าเลขจำนวนจริงใดๆ มีค่าน้อยกว่า .000000000000000000000000000000000000003 (เลข '0' 38 ตัว) ค่าของมันจะถูกเปลี่ยนเป็น 0

3.10 การแสดงจำนวนเลขบนจอ

แม้ว่า "ไทยปกาสิต" จะสามารถแทนค่าเลขจำนวนจริงทั้งบวกและลบได้ถึง 39 หลัก ดังกล่าวในข้อ 3.9 ก็ตาม แต่การแสดงจำนวนเลขบนจอจะแสดงด้วยตัวเลขเพียง 9 หลัก เท่านั้น โดยทำการปัดเศษส่วนที่เกิน 9 หลัก และแสดงจำนวนเลขที่เกิน 9 หลัก ด้วยวิธีเขียนจำนวนเลขแบบวิทยาศาสตร์ ซึ่งตามปกติจะเขียนเป็น $\pm N X 10^{\pm M}$ โดยที่ N เป็นเลขระหว่าง 1-9 และ M เป็นเลขระหว่าง1 ถึง 38 แต่บนจอจะปรากฏเป็น ±N E ±M หรืออีกนัยหนึ่งใช้อักษร E (ย่อมาจากคำว่า EXPONENT) แทน XIO และลดระดับ ±M ลงมาเขียนบนบรรทัดเดียวกันนั่นเอง

ตัวอย่างการเขียนจำนวนเลขแบบวิทยาศาสตร์ที่ปรากฏบนจอ มีดังนี้

เขียนแบบธรรมดา	**เขียนแบบวิทยาศาสตร์**
1000000000	1E + 09
.0000000001	1E - 09
200	2E + 02
-123456789	-1.23456789 E + 09
-.00000123456789	-1.23456789E - 06

ตัวอย่างต่อไปนี้จะแสดงการปัดเศษ และการเปลี่ยนจำนวนเลขที่มีจำนวนหลักเกิน 9 หลัก ให้เข้าสู่วิธีการเขียนแบบวิทยาศาสตร์

จงทดลองทำตาม

พิมพ์ 1234567891

1.23456789E + 09

พิมพ์ -123456789123456789

1.23456789E + 17

พิมพ์ -150000475.75

-150000476

พิมพ์ 900000000.7558

900000000.8

พิมพ์ .1234567891

.123456789

พิมพ์ -123456789123456789

-1.23456789E + 17

พิมพ์ 00000000009000000007558

9.000000008E - 10

หมายเหตุ ข้อนี้อาจยากเกินไปสำหรับผู้ที่ไม่มีพื้นฐานทางวิทยาศาสตร์ถึงระดับมัธยมปลายขึ้นไป ในกรณีเช่นนี้ให้ข้ามไปอ่านหัวข้อถัดไป

3.11 ตัวแปร

ตัวแปร คือ ชื่อของค่าตัวเลขที่เราจะใช้ในการคำนวณ ยกตัวอย่าง เช่น

ค่าใช้จ่าย = ค่าอาหาร + ค่ารถ

ในกรณีที่เราต้องการคำนวณ**ค่าใช้จ่าย**รายวัน ซึ่งแต่ละวันจะมีค่าตัวเลขต่างกัน หากเราใช้ตัวเลขโดยตรง เราจะต้องเป็นผู้กำหนดขั้นตอนการคำนวณเองเช่น

วันแรก ค่าอาหาร = 20 บาท ค่ารถ = 10 บาท

เราต้องจิ้มเครื่องคิดเลข ดังนี้ 20 + 10 = (อ่านผลลัพธ์ = 30)

วันที่ 2 ค่าอาหาร = 18 บาท ค่ารถ = 25 บาท

เราต้องจิ้มเครื่องคิดเลขในลักษณะเดียวกัน

ถ้ามีตัวเลขที่จะต้องคำนวณสัก 90 ชุด เราคงอยากมีวิธีการที่สามารถป้อนแต่ตัวเลขโดยให้เครื่องคิดเลขทำการคำนวณเอง ตามขั้นตอนที่กำหนดไว้ล่วงหน้า ซึ่งหลักการนี้ ก็คือ **การเขียนโปรแกรม** สำหรับคอมพิวเตอร์นั่นเอง แต่ก่อนที่เราจะสามารถทำเช่นนั้นได้ เราต้องมีชื่อแทนจำนวนต่างๆ ที่จะนำไปคำนวณเสียก่อน โดยชื่อแต่ละชื่อจะมีค่าตามที่กำหนดในการคำนวณแต่ละครั้ง **ค่าใช้จ่าย** ค่าอาหาร

ค่ารถ ก็เป็นชื่อที่มีลักษณะเช่นนี้ ซึ่งเราเรียกว่า **ตัวแปร** **สมการ** ก็คือสูตรคำนวณซึ่งแสดงว่าจะต้องนำค่าตัวแปรไปทำการคำนวณอย่างไร ในตัวอย่างของเราสมการบอกว่าให้นำ **ค่าอาหาร** มาบวกกับ **ค่ารถ** ผลลัพธ์ที่ได้ก็คือ **ค่าใช้จ่าย**

เนื่องจากคอมพิวเตอร์ทำงานเกี่ยวกับตัวเลขและข้อความ (ตัวอักขระ) ตัวแปรจึงมี 2 ประเภท คือ

1. **ตัวแปรเลข** ได้แก่ตัวแปรที่แทนตัวเลข
2. **ตัวแปรอักขระ** ได้แก่ตัวแปรที่แทนตัวอักษรหรือข้อความ

หลักการตั้งชื่อของตัวแปรในภาษา"ไทยปกาสิต" มีดังนี้

ตัวแปรอาจประกอบด้วยตัวอักขระ 1,2 หรือ 3 ตัว

[1] [2] [3]

[1] ตัวแรก ต้องเป็น **อักษร A-Z**

[2] ตัวที่สองเป็น **อักษร** A-Z หรือ **ตัวเลข** ก็ได้ <u>หรือไม่มีก็ได้</u>

[3] ตัวที่สาม เป็นตัวบอก **ชนิดของตัวแปร** ดังนี้

$ แสดงว่าเป็น **ตัวแปรอักขระ**

% แสดงว่าเป็น **ตัวแปรเลข ที่เป็นเลขจำนวนเต็ม**)

ถ้าไม่ระบุ (ไม่มีตัวที่สาม) แสดงว่าเป็น **ตัวแปรเลข** ที่เป็น **เลขจำนวนจริง**

ตัวอย่างชื่อตัวแปรที่ใช้ได้มีดังนี้

ชื่อ	**ชนิด**
A%	เลขจำนวนเต็ม
A1	เลขจำนวนจริง
A1%	เลขจำนวนเต็ม
AA	เลขจำนวนจริง
XI%	เลขจำนวนเต็ม
A$	อักขระ
MN$	อักขระ
F6$	อักขระ

ตัวอย่างชื่อตัวแปรที่ต้องห้ามมีดังนี้

ชื่อ	**เหตุที่ต้องห้าม**
3	ไม่ขึ้นต้นด้วยอักษร A-Z
4B	ไม่ขึ้นต้นด้วยอักษร A-Z
A#	มีตัวที่ 2 ไม่ใช่ตัวอักษร A-Z หรือตัวเลข
31%	ไม่ขึ้นต้นด้วยอักษร A-Z
3D%	ไม่ขึ้นต้นด้วยอักษร A-Z
3$	ไม่ขึ้นต้นด้วยอักษร A-Z
3D$	ไม่ขึ้นต้นด้วยอักษร A-Z

3.12 ค่าของตัวแปร

ค่าของ **ตัวแปรเลข** จะต้องเป็นตัวเลขภายในขอบเขตดังนี้

จำนวนเต็ม	-32767	ถึง	+32767
จำนวนจริง	-1E 38	ถึง	+1E 38

ค่าของ**ตัวแปรอักขระ** อาจเป็นขบวนอักขระใดๆ (ตัวอักษรไทย, อังกฤษ, ตัวเลข และเครื่องหมายใดๆ รวมทั้งเว้นวรรค) แต่ต้องมีความยาวไม่เกิน 255 ตัว

3.13 คำสงวน

คำสงวน คือ คำที่ "ไทยปกาสิต" ใช้เป็นคำสั่ง และห้ามนำมาใช้ตั้งชื่อ ตัวแปร หรือส่วนประกอบของชื่อตัวแปร แต่อนุญาตให้มีคำเหล่านี้ได้ในข้อความที่อยู่ในเครื่องหมายอัญญประกาศ

บัญชีคำสงวนดูได้จากตาราง ในภาคผนวก ผ.3

ตาราง 3.1 คำสั่งภาษาไทย เทียบกับ คำสั่งภาษาอังกฤษ

APPLESOFT	ไทยปกาสิต	APPLESOFT	ไทยปกาสิต	APPLESOFT	ไทยปกาสิต
ABS	สัมบูรณ์	HCOLOR=	ภลสี	ON	ขึ้นกับ
AND	และ	HGR	ภล	ONERR	เมื่อผิด
ASC	รหัส	HGR2	ภลต	OR	หรือ
AT	ที่	HIMEM	ขอบจำบน	PDL	คันบังคับ
ATN	ATN	HLIN	เส้นนอน	PEEK	หยิบ
CALL	เรียก	HOME	ล้างจอ	PLOT	จุด
CHR$	อักขร	HPLOT	ภลจุด	POKE	หยอด
CLEAR	ล้าง	HTAB	ย้ายราบ	POP	โดด
COLOR=	สี	IF	ถ้า	POS	ตำแหน่ง
CONT	ทำต่อ	IN#	เข้า	PRINT	พิมพ์
COS	COS	INPUT	รับ	PR#	ช่อง
DATA	ข้อมูล	INT	INT	READ	อ่าน
DEF	กำหนด	INVERSE	สลับแสง	RECALL	เรียกคืน
DEL	ตัด	LEFT$	ซ้าย	REM	'
DIM	ขนาด	LEN	ความยาว	RESTORE	สงวน
DRAW	วาด	LET	ให้	RESUME	เดินต่อ
END	จบ	LIST	แจง	RETURN	กลับ
EXP	EXP	LOAD	บรรจุ	RIGHT$	ขวา
FLASH	กระพริบ	LOG	LOG	RND	สุ่ม
FN	สูตร	LOMEM	ขอบจำล่าง	ROT=	หมุน
FOR	จาก	MID$	กลาง	RUN	ทำ
FRE	ว่าง	NEW	ใหม่	SAVE	บันทึก
GET	จับ	NEXT	เพิ่ม	SCALE=	มาตรา
GOSUB	ไปภาค	NORMAL	ปกติ	SCRN(	สีจอ
GOTO	ไป	NOT	ไม่	SGN	เครื่องหมาย
GR	ภย	NOTRACE	เลิกตาม	SHLOAD	บรรจุภาพ

APPLESOFT	ไทยปกาสิต	APPLESOFT	ไทยปกาสิต	APPLESOFT	ไทยปกาสิต
SIN	SIN	TAB(	เลื่อน(	VAL	ค่า
SPC(	เว้นวรรค	TAN	TAN	VLIN	เส้นดิ่ง
SPEED=	ความเร็ว	TEXT	จอปกติ	VTAB	ย้ายดิ่ง
SQR	SQR	THEN	แล้ว	WAIT	รอ
STEP	ทีละ	TO	ถึง	XDRAW	วาดสี
STOP	หยุด	TRACE	ตาม		
STORE	เก็บ	USR	เรียกใช้		
STR$	แปลง				

บทที่ 4
การเขียนโปรแกรม ด้วย "ไทยปกาสิต"

4.1 สิ่งที่จะต้องรู้ในการเขียนโปรแกรม

ในการเขียนโปรแกรมสำหรับคอมพิวเตอร์ ไม่ว่าจะเขียนด้วยภาษาใด สิ่งที่ผู้เขียนจะต้องรู้ คือ

1. คำสั่ง และวิธีใช้คำสั่ง
2. กฎเกณฑ์หรือไวยากรณ์ของภาษานั้น
3. หลักการของโปรแกรม หรืออีกนัยหนึ่งก็ คือ การวางแผนให้คอมพิวเตอร์ทำงานตามที่เราต้องการ

ข้อ 1 และ 2 เป็นสิ่งที่สามารถเรียนรู้ได้โดยตรง แต่ข้อ 3 นอกจากความรู้ซึ่งศึกษาจากตำรับตำรา จากตัวอย่างที่พบเห็นแล้ว ยังต้องอาศัยอีกสิ่งหนึ่ง ซึ่งเป็นศิลปะเฉพาะตัวบุคคลด้วย ซึ่งแต่ละคนอาจมีไม่เท่ากันแต่ทุกคนก็สามารถจะพัฒนาขึ้นมาได้โดยอาศัยประสบการณ์

ในบทนี้จะได้แนะนำการเขียนโปรแกรมโดยอาศัยตัวอย่างการใช้คำสั่งแต่ละคำพร้อมทั้งอธิบายกฎเกณฑ์หรือไวยกรณ์ในส่วนที่เกี่ยวข้องด้วย

4.2 พิมพ์

วิธีใช้คำสั่ง **พิมพ์** ได้อธิบายไว้แล้วในข้อ 3.3, 3.4 และ 3.5 เพื่อเป็นการทบทวน จงทำตามตัวอย่างต่อไปนี้ แล้วบันทึกผลที่ปรากฏบนจอในบรรทัดที่เว้นไว้ให้

พิมพ์ข้อความ

พิมพ์ "บทที่ 4"

....

พิมพ์ "การเขียนโปรแกรม" , "ด้วยไทยปกาสิต"

....

พิมพ์ตัวเลข

พิมพ์ 1320

....

พิมพ์ 1320, 1460, 1580, 1660, 1750, 1860

....

....

พิมพ์ 1320; 1460

....

พิมพ์ผลลัพธ์การคำนวณ

พิมพ์ 123 + 234 - 56

....

พิมพ์ 13 * 12

....

พิมพ์ 125/4

....

พิมพ์ 8 3

....

4.3 การกำหนดค่าตัวแปร

จงพิมพ์คำสั่งต่อไปนี้

```
▷    ใหม่
10   A =  2
20   B =  3
30   C =  A * B
40   พิมพ์ C
▷    ทำ
6
```

คำสั่งบรรทัด 10 เป็นคำสั่งที่ใช้กำหนดค่าตัวแปร A ให้เท่ากับ 2 คำสั่งบรรทัด 20 กำหนดให้ B เท่ากับ 3 คำสั่งบรรทัด 30 กำหนดให้ C มีค่าเท่ากับผลคูณของ A กับ B จะเห็นได้ว่าการกำหนดค่าตัวแปร อาจกำหนดให้มีค่าเท่ากับค่าคงที่ หรือกำหนดให้มีค่าขึ้นอยู่กับตัวแปรอื่น ตามสูตรคณิตศาสตร์ก็ได้

หอสมุดแห่งชาติ

4.4 การใช้หมายเหตุ

ในกรณีที่เขียนโปรแกรมที่มีความสลับซับซ้อนมาก เราควรใช้หมายเหตุกำกับเพื่อให้ผู้เขียนเองและผู้อื่นที่มาอ่านโปรแกรมทราบว่าแต่ละขั้นตอนของโปรแกรมทำอะไรบ้าง หมายเหตุคือสิ่งที่คอมพิวเตอร์ไม่ถือว่าเป็นคำสั่ง จึงไม่มีการปฏิบัติตาม เราใช้เครื่องหมาย ' นำหน้าข้อความที่จะใช้เป็นหมายเหตุ ข้อความนี้อาจอยู่ในบรรทัดต่างหาก หรือต่อท้ายคำสั่งอื่นในบรรทัดเดียวกันโดยใช้ : คั่นก็ได้ เช่น

```
10  ' โปรแกรมแสดงการกำหนดค่าตัวแปร
20  A = 2 : ' กำหนดให้ A เป็น 2
```

4.5 รับ และ จับ

โปรแกรมในข้อ 4.3 เป็นโปรแกรมหาผลคูณของเลขสองจำนวน แล้วพิมพ์คำตอบ แต่โปรแกรมนี้กำหนดเลขสองจำนวนไว้ตายตัว เป็น 2 กับ 3 ถ้าเราต้องการให้โปรแกรมสามารถทำการคูณเลข 2 จำนวนใด ๆ ก็ได้ เราจะแก้ไขใหม่เป็นดังนี้

```
▷   ใหม่
10  รับ A
20  รับ B
30  C = A * B
40  พิมพ์ C
▷   ทำ
?   7
?   6
42
```

คำสั่งบรรทัด 10 สั่งให้คอมพิวเตอร์รอรับค่า A จากแป้นพิมพ์ คอมพิวเตอร์จะแสดงเครื่องหมายคำถามออกมา เพื่อรอให้เราตอบ พอเรากด 7 RETURN คอมพิวเตอร์จะรับเลข 7 เข้าไป แล้วจะมาทำตามคำสั่งบรรทัด 20 ฯลฯ

ตามตัวอย่างข้างบน สมมติว่า A เป็นราคาสินค้า และ B เป็นจำนวนสินค้า ถ้าเราต้องการให้คอมพิวเตอร์บอกเราว่าต้องค่าใด เราก็สามารถทำได้โดยแก้บรรทัด 10, 20 เป็นดังนี้

```
10  รับ "ราคา = ";A
20  รับ "จำนวน = "; B
```

คราวนี้ลองสั่ง **ทำ** ใหม่ ผลจะเป็นดังนี้

▷ ทำ

ราคา = □

เครื่องหมายคำถามหายไป แต่มีคำว่า "ราคา=" ปรากฏขึ้น ขณะนี้คอมพิวเตอร์รอให้เราพิมพ์ค่าตัวเลขของ **ราคา** ที่ตำแหน่งของเคอร์เซอร์ เมื่อพิมพ์เสร็จให้กด **รีเทอน** คอมพิวเตอร์จะรอรับค่าตัวเลขของ **จำนวน** ต่อไป

ทดลองสั่ง**ทำ** หลาย ๆ ครั้ง โดยใช้ตัวเลขต่าง ๆ กัน หลักเดียวบ้าง 2-3 หลัก บ้าง

คราวนี้ลองเปลี่ยนบรรทัด 10, 20 ใหม่ เป็น

```
10  จับ A
20  จับ B
```

ลองสั่งทำดู โดยใช้ตัวเลขต่าง ๆ กัน จะพบว่า **จับ** ทำงานคล้าย ๆ **รับ** แต่ต่างกันคือ ตัวเลขที่เราพิมพ์จะไม่ปรากฏบนจอ พอเรากดตัวเลข 1 ตัว คอมพิวเตอร์จะรับเอาไปเลย โดยไม่รอให้เรากด **รีเทอน** เราจึงไม่สามารถใช้ **จับ** กับเลข หลายหลักได้

คำสั่ง **จับ** เป็นคำสั่งบอกให้คอมพิวเตอร์คอยจับดูว่ามีการพิมพ์อักษรหรือเลขตัวใดเข้ามาทางแป้นพิมพ์ ถ้ามีก็ให้รับอักษรหรือเลขนั้นไปเลย

จงทดลองคำสั่งต่อไปนี้

```
100  จับ A$
110  พิมพ์ A$
```

▷ ทำ 100

คำสั่งนี้สั่งให้เริ่มทำตั้งแต่บรรทัด 100

ที่นี้ถ้าเรากดแป้นใด อักษรนั้นก็จะปรากฏบนจอ

ประโยชน์ของ **จับ** คือใช้ในกรณีที่เราต้องการให้คอมพิวเตอร์ ปฏิบัติตามคำสั่งที่กำหนด ทันทีที่เรากดแป้นที่กำหนด

จงพิมพ์คำสั่งบรรทัด 100, 110 ใหม่ เป็นดังนี้

```
100   พิมพ์ "ต้องการทำต่อหรือไม่? (ท = ทำ, ม = ไม่)
110   จับ A$
120   ถ้า A$ = "ท" แล้วไป 10
130   ถ้า A$ = "ม" แล้ว จบ
140   ไป 110
```

ลองสั่ง **ทำ** ดู เครื่องจะถามเราว่า "ต้องการทำต่อหรือไม่?" ถ้าเราตอบว่า ท (ทำ) คอมพิวเตอร์จะวนกลับไปทำบรรทัด 10 ใหม่ ถ้าเราตอบว่า ม (ไม่) คอมพิวเตอร์จะออกจากโปรแกรม

การทำงานของคำสั่งตั้งแต่บรรทัด 120, 130 จะอธิบายในหัวข้อต่อไป

4.6 ถ้า ...แล้ว และ ไป

คำสั่ง **ถ้า...แล้ว** เป็นคำสั่งให้ตรวจสอบเงื่อนไขตามตัวอย่าง คือ ถ้าตัวอักขระ A$ ที่**จับ** มาได้ คือ "ท" ก็ให้ทำส่วนที่เหลือของบรรทัด 120 นั่นคือ ไป 10 แต่ถ้าไม่ใช่ให้ข้ามไปทำบรรทัดต่อไป

ในทำนองเดียวกัน บรรทัด 130 จะตรวจสอบว่า A$ ที่ **จับ** มาได้คือ "ม" ใช่หรือไม่ ถ้าใช่ก็ให้ทำตามคำสั่ง "จบ" คือออกจากโปรแกรม แต่ถ้าไม่ใช่ก็ให้ข้ามไปทำบรรทัดต่อไป

พอมาถึงบรรทัด 140 แสดงว่า A$ ที่จับมาได้ ไม่ใช่ทั้ง "ท" และ "ม" แสดงว่าผู้ตอบ ตอบไม่ตรงคำถาม จึงให้กลับไปรอรับคำตอบใหม่ที่บรรทัด 110

คำสั่ง **ไป** ในบรรทัด 120 และ 140 เป็นคำสั่งให้ไปทำต่อที่เลขบรรทัดนั้น ๆ เราอาจใช้คำสั่งไป ทำให้โปรกแรมวนกลับไปตั้งต้นใหม่อย่างไม่รู้จบได้ โดยการเติมบรรทัด 50 ดังนี้

```
50 ไป 10
```

ทุกครั้งที่ทำมาถึงบรรทัด 50 โปรแกรมจะวนกลับไปเริ่มที่บรรทัด 10 ฉะนั้นโปรแกรมตั้งแต่บรรทัดที่ 100 ถึง 140 จึงไม่มีการ **ทำ** เลย และโปรแกรมที่กลายเป็นโปรแกรมไม่รู้จบนี้ จะมีวิธีออกจากโปรแกรมได้วิธีเดียวคือการ RESET

4.7 จาก...เพิ่ม

การใช้คำสั่ง **ไป** ให้กลับไปที่ต้นโปรแกรม ทำให้โปรแกรมวนกลับไปตั้งต้นใหม่อย่างไม่รู้จบ แต่ถ้าเราต้องการให้วนทำเพียงจำนวนครั้งที่ต้องการ เราจะใช้คำสั่ง **จาก...เพิ่ม** ดังตัวอย่างต่อไปนี้

```
▷   ใหม่
10  จาก N = 1 ถึง 10
20  พิมพ์ N, อักขร (64 + N)
30  เพิ่ม N
40  จบ
▷   ทำ
1            A
2            B
3            C
4            D
5            E
6            F
7            G
8            H
9            I
10           J
▷
```

โปรแกรมจะพิมพ์ตัวอักษรภาษาอังกฤษ 10 ตัวแรก ทั้งนี้เพราะรหัสมาตรฐาน (ASCII CODE) ของตัวอักษรภาษาอังกฤษ เริ่มที่ อักขร (65) = A จนถึงอักขร (90) = Z

การทำงานของคำสั่ง **จาก...เพิ่ม** เป็นดังนี้

บรรทัด 10 คอมพิวเตอร์จะกำหนดค่า N = 1 และกำหนดค่า MAX = 10

บรรทัด 20 พิมพ์ค่า N คือ 1 และพิมพ์อักขร (65) คือ A

บรรทัด 30 กำหนดค่า N = N + 1 = 2 แล้วกลับไปบรรทัด 10

บรรทัด 10 ทดสอบว่า N > MAX หรือไม่ ถ้าใช่กระโดดไปทำบรรทัดที่

ตามหลังคำสั่ง **เพิ่ม** คือ 40 จบ ถ้า N > MAX ไม่เป็นความจริง จะทำบรรทัด 20 ต่อ

บรรทัด 20 พิมพ์ค่า N คือ 2 และพิมพ์อักขร (66) คือ B

บรรทัด 30 กำหนดค่า N = N + 1 = 3 แล้วไปทำบรรทัด 10 อีก

วนเช่นนี้เรื่อยไป จนถึงรอบสุดท้าย เมื่อ N = 10 หลังจากพิมพ์ค่า N และอักขร (74) แล้ว จะเพิ่ม N เป็น 11 เมื่อกลับไปทดสอบในบรรทัดที่ 10 พบว่า N > MAX ถ้าจะกระโดดไป 40 คือ จบโปรแกรม

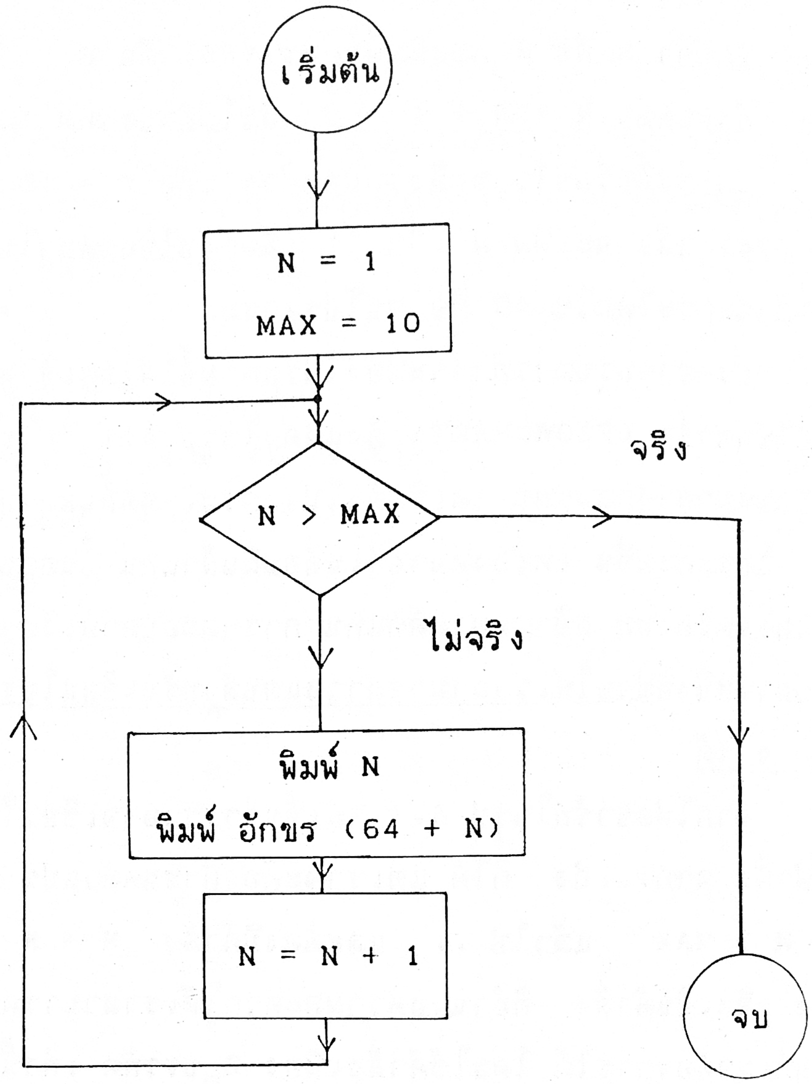
เราอาจสรุปการทำงานของโปรแกรมนี้ได้โดยเขียนเป็นแผนภูมิชนิดหนึ่ง เรียกว่า โฟลชาร์ท (FLOWCHART) ดังแสดงในรูป 4.1 ในการเขียนโฟลชาร์ท เราใช้เครื่องหมายวงกลมแทน จุดเริ่มต้นโปรแกรม, จุดสิ้นสุดโปรแกรม และจุดต่อเชื่อมกับ โปรแกรมอื่น เครื่องหมายสี่เหลี่ยมผืนผ้าแทน ขั้นตอนการทำงานของคอมพิวเตอร์ และเครื่องหมายข้าวหลามตัดแทน การทดสอบตามเงื่อนไข โฟลชาร์ทเป็นเครื่องมืออย่างหนึ่งที่ช่วยให้เราสามารถวางแผนสำหรับเขียนโปรแกรมที่มีความสลับซับซ้อนมาก ๆ ได้

จากโฟลชาร์ทในรูป 4.1 จะเห็นว่าเราอาจเขียนโปรแกรมข้างต้นใหม่ โดยไม่ใช้คำสั่ง **จาก...ถึง** ก็ได้ แต่เราจะต้องกำหนดตัวแปร MAX = 10 ต้องมีคำสั่ง **ถ้า N > MAX แล้วไป...** และต้องมีคำสั่ง **N = N + 1** ดังนั้นคำสั่ง **จาก...ถึง** จึงเป็นคำสั่ง ที่อำนวยความสะดวกให้เราสามารถสั่งให้วนทำงานตามจำนวนรอบที่เราต้องการได้ โดยใช้คำสั่งเพียง 2 บรรทัด เท่านั้น

จงทดลองเขียนโปรแกรมเดียวกันนี้ โดยไม่ใช้ **จาก...ถึง** เพื่อให้แน่ใจว่าจะได้ผลอย่างเดียวกัน

4.8 ไปภาค และ กลับ

ภาค คือ ส่วนย่อยที่แบ่งมาจากส่วนใหญ่ โปรแกรมขนาดใหญ่ ควรแบ่งออกเป็นหลายๆ ภาค เพื่อความสะดวกในการเขียนและแก้ไขโปรแกรม เราจะเขียนและแก้ไขโปรแกรมในแต่ละภาคได้โดยอิสระจากกัน นอกจากนี้งานที่มีการทำซ้ำ ๆ กันบ่อยๆ ถ้าเราแยกเฉพาะงานนั้นออกเป็นภาคต่างหาก เวลาจะให้คอมพิวเตอร์ทำงานนั้น เราก็ใช้คำสั่ง **ไปภาค** (ต่อด้วยเลขบรรทัดที่เป็นจุดเริ่มต้นของภาค) บรรทัด สุดท้ายของภาคจะเป็นคำสั่ง **กลับ** เสมอ ซึ่งจะส่งการควบคุมกลับมาที่บรรทัดถัดจากคำสั่ง **ไปภาค** ที่เพิ่งทำเสร็จแล้วนั้น



รูป 4.1 โฟลว์ชาร์ท แสดงการทำงานของ โปรแกรมที่ใช้ <u>จาก</u> ... <u>ถึง</u>

```
**************
**          **
** ไทยปกาสิต **
**          **
**************
```

รูป 4.2 ภาพที่ต้องการพิมพ์บนจอ

ยกตัวอย่าง เช่น ถ้าเราต้องการพิมพ์ภาพในรูป 4.2 บนจอ เราอาจเขียนโปรแกรมดังนี้

```
▷    ใหม่
10   ไปภาค 100
20   ไปภาค 150
30   ไปภาค 200
40   ไปภาค 150
50   ไปภาค 100
60   จบ
100  พิมพ์ "          **************"
      (เว้นวรรค 10 ตัว)  ( 14 ตัว)
110  กลับ
150  พิมพ์ "          **          **"
160  กลับ
200  พิมพ์ "          ** ไทยปกาสิต **"
210  กลับ
```

โปรแกรมนี้จะประหยัดเวลาให้เราได้ โดยเราไม่ต้องพิมพ์บรรทัดยากๆ ที่ไปซ้ำกันมากกว่า 1 ครั้ง อย่าลืมใส่ **จบ** ที่บรรทัด 60 เพราะมิฉะนั้น พอทำ 50 เสร็จแล้วโปรแกรมจะเลยไปทำ 100 และจะพบคำสั่ง **กลับ** ที่บรรทัด 11โดย ไม่มีคำสั่ง **ไปภาค** ที่คู่กัน เครื่องจะรายงานความผิดพลาดว่า

? **มีกลับไม่มีไปภาค**

4.9 การใช้คำสั่ง **จาก.....เพิ่ม** ซ้อนกันหลายชั้น

เราสามารถใช้คำสั่ง **จาก...เพิ่ม** ซ้อนกันหลายชั้นได้ แต่มีหลักอยู่ว่า

1. **จาก** ทุกตัวจะต้องมี**เพิ่ม** คู่กันเสมอ และ
2. ถ้าเราเขียนเส้นโยงระหว่าง**จาก** กับ**เพิ่ม** แต่ละคู่ เส้นโยงของคำสั่งต่างคู่กันจะต้องไม่ตัดกัน เช่น

```
10   จาก M = 1 ถึง 5
20   จาก N  = 1 ถึง 10
```

```
30    ...
40    จาก I = 1 ถึง 7
50    ...
60    เพิ่ม I
70    ...
80    เพิ่ม N
90    เพิ่ม M
```

4.10 การใช้คำสั่งไปภาค **ซ้อนกันหลายชั้น**

เราสามารถใช้คำสั่ง **ไปภาค** ซ้อนกันหลายชั้น กล่าวคือ ภาคใดภาคหนึ่งของโปรแกรม ยังสามารถเรียกภาคอื่นของโปรแกรมได้อีก ทั้งนี้ไม่มีกฎเกณฑ์พิเศษอะไร เพียงแต่ว่าคำสั่งสุดท้ายของทุก**ภาค** ต้องเป็น**กลับ** เพื่อให้ส่งกลับมาทำงานต่อได้อย่างถูกต้องเท่านั้น

ข้อห้ามเพียงข้อเดียวของการใช้คำสั่ง**ไปภาค** ซ้อนกัน คือ ห้ามมิให้ภาคใดภาคหนึ่งสั่งให้**ไปภาค** ของตัวเองไม่ว่าโดยตรงหรือโดยอ้อมก็ตาม

4.11 **ขึ้นกับ** A...ไป และ **ขึ้นกับ** A...ไปภาค

คำสั่งคู่นี้ ทำให้เราสามารถตั้งเงื่อนไขให้เกิดการเลือกปฏิบัติได้ โดยขึ้นอยู่กับค่าตัวแปร A ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1

```
▷    ใหม่
10   ขึ้นกับ A ไป 100, 200, 300
20   รับ A
30   ถ้า A < 1 หรือ A > 3 ไป 20
100  พิมพ์ "ทำบรรทัด 100" : ไป 10
200  พิมพ์ "ทำบรรทัด 200" : ไป 10
300  พิมพ์ "ทำบรรทัด 300" : ไป 10
```

ลองพิมพ์โปรแกรมนี้ และลองทำ ดูหลายๆ ครั้ง โดยป้อนค่า A ต่างๆ กัน

ตัวอย่างที่ 2

```
▷ ใหม่
10  ขึ้นกับ DAY ไปภาค 100, 200, 300, 400, 500, 600, 700
20  รับ DAY
30  ถ้า DAY < 1 หรือ DAY > 7 ไป 20
100  พิมพ์ " วันอาทิตย์สีแดง" : กลับ
200  พิมพ์ " วันจันทร์สีเหลือง" : กลับ
300  พิมพ์ " วันอังคารสีชมภู" : กลับ
400  พิมพ์ " วันพุธสีเขียว" : กลับ
500  พิมพ์ " วันพฤหัสสีแสด" : กลับ
600  พิมพ์ " วันศุกร์สีฟ้า" : กลับ
700  พิมพ์ " วันเสาร์สีม่วง": กลับ
```

แม้ว่าตัวอย่างข้างต้นจะเป็นตัวอย่างที่ค่อนข้างจะไร้สาระ แต่ก็ทำให้สามารถเข้าใจถึงพลังของคำสั่ง**ขึ้นกับ...ไปภาค** ได้ ถ้าเราไม่ลืมว่าในแต่ละภาคสามารถมีคำสั่งให้ทำงานแต่ละอย่างเป็นสิบหรือร้อยบรรทัดก็ได้

4.12 **อ่าน** และ **ข้อมูล**

ในตัวอย่างก่อนๆ เราสามารถป้อนข้อมูลแก่คอมพิวเตอร์ได้สองวิธี คือ วิธีแรก โดยการกำหนดค่าตัวแปร เช่น

A = 5

B = 3

DAY = 7

NANE$ = "วัลลภ" เป็นต้น

วิธีที่สอง โดยการใช้คำสั่ง **รับ** เช่น รับ DAY ซึ่งในที่นี้ DAY เป็นตัวแปรเลขมีค่า 1 ถึง 7 หรือ รับ NAME$ โดยที่ NAME$ เป็นตัวอักขร เป็นต้น

วิธีทั้งสองดังกล่าวใช้การได้ดี ถ้า

(ก) มีจำนวนข้อมูลไม่มาก

(ข) ไม่ต้องการทำโปรแกรมซ้ำๆ โดยเปลี่ยนข้อมูลใหม่ทั้งชุดทุกครั้ง

ถ้าเงื่อนไขตาม (ก) หรือ (ข) ไม่เป็นจริง การป้อนข้อมูลด้วยคำสั่ง **อ่าน** และ **ข้อมูล** จะได้ผลดีกว่า

การใช้คำสั่ง**อ่าน** และ**ข้อมูล** มีหลักกดังนี้

1. ตามปรกติ มักเขียนคำสั่ง**อ่าน** แทรกไว้ตามตำแหน่งที่ต้องการเรียกใช้ข้อมูล และเขียนคำสั่ง**ข้อมูล** ไว้ท้ายโปรแกรมเพื่อให้แก้ไขเปลี่ยนแปลงได้ง่าย แต่ไม่เป็นการบังคับว่าจะต้องทำเช่นนี้

2. เมื่ออ่านข้อมูลตัวใดไปแล้ว ข้อมูลตัวนั้นจะถูกทิ้งไป ในการอ่านครั้งต่อไป จะอ่านข้อมูลตัวถัดไป

3. เมื่อพบคำสั่ง**อ่าน** (ตามด้วยชื่อตัวแปร) คอมพิวเตอร์ จะนำข้อมูลตัวแรก ที่ยังใช้การอยู่มากำหนดค่าเป็นค่าของตัวแปรนั้นๆ

4. จำนวนตัวแปรที่ตามหลังคำสั่งอ่าน จะต้องไม่มากกว่าจำนวนข้อมูลที่ตามหลังคำสั่งข้อมูล มิฉะนั้นจะรายงานความผิดพลาดว่า **"หมดข้อมูล"**

ตัวอย่าง เช่น

```
220   อ่าน   A, B, C          (A =  10)
...                           (B =  20)
...                           (B =  30)
340   อ่าน   C, D             (C =  40)
...                           (D =  50)
490   อ่าน   A, E, F, G       (A =  60)
                              (E =  70)
                              (F =  80)
                              (G =  90)
500   อ่าน   B                (B = 100)
1000  ข้อมูล  10, 20, 30,  40, 50,  60, 70, 80, 90,  100
               (220)        (340)     (490)          (500)
```

หมายเหตุ เราอาจเขียนบรรทัด 1000 แยกเป็นหลายบรรทัดก็ได้ เช่น

```
1000  ข้อมูล  10, 20, 30, 40
1010  ข้อมูล  50, 60, 70
1020  ข้อมูล  80,90,100
```

4.13 **สงวน**

คำสั่ง **สงวน** เป็นคำสั่งที่เก็บข้อมูลที่ทิ้งไปแล้ว (หลังจากอ่าน) กลับมาใช้ใหม่ เราจะใช้คำสั่ง **สงวน** เมื่อใดก็ได้ ผลของการใช้คำสั่งนี้ก็คือ การอ่านข้อมูลครั้งต่อไป จะกลับไปเริ่มอ่านที่ข้อมูลตัวแรกที่ตามหลังคำสั่ง **ข้อมูล** (แม้ว่าข้อมูลตัวนั้นจะเคยถูกอ่านมาแล้วก็ตาม)

4.14 **ตัวแปรแถว**

ตัวแปรแถวอาจเปรียบเทียบได้กับตึกแถว ซึ่งทุกคูหามีขนาดเท่ากัน และวางแปลนอย่างเดียวกัน แต่ทว่า แต่ละคูหาอาจประกอบกิจการต่างกัน เช่นเป็นร้านขายอาหาร ร้านขายเสื้อผ้า ร้านเสริมสวย เป็นต้น ตัวแปรแถว ก็คือตัวแปรกลุ่มหนึ่งซึ่งมีคุณสมบัติบางอย่างร่วมกัน แต่คุณสมบัติปลีกย่อยต่างกัน เช่น ถ้าเรารวบรวมข้อมูลเกี่ยวกับนักเรียน 5 คนมาเขียนเป็นตาราง ดังนี้

ลำดับที่ I	**ชื่อ N$(I)**	**อายุ A(I)**	**จำนวนพี่น้อง K(I)**
1	ดำ	10	1
2	แดง	11	2
3	เปี๊ยก	12	3
4	น้อย	11	2
5	นิด	10	2

เพื่อความสะดวกในการเรียกชื่อตัวแปร เราจึงใช้ชื่อตัวแปรเดียวกันสำหรับข้อมูลชนิดเดียวกัน แต่ให้มีเลขลำดับกำกับ เช่น

N$(I) หมายถึง <u>ชื่อ</u> ลำดับที่ I (I = 1 ถึง 5)

A(I) หมายถึง <u>อายุ</u> ลำดับที่ I (I = 1 ถึง 5)

K(I) หมายถึง <u>จำนวนพี่น้อง</u> ลำดับที่ I (I = 1 ถึง 5)

เช่นเดียวกับตึกแถวซึ่งอาจมีหลายชั้น ตัวแปรแถวก็อาจมีหลายชั้นได้ จะขอยกตัวอย่างตัวแปรแถว 2 ชั้น (Two-dimensional array) ดังนี้

สมมติข้อมูลชุดที่แล้ว มาจากการสุ่มตัวอย่างของนักเรียนชั้น ป.5 คราวนี้เราจะสำรวจเพิ่มเติม โดยการสุ่มตัวอย่าง นักเรียนชั้น ป.6 และ ป.7 อีก

ชั้นละ 5 คน เราจะมีตารางแบบเดียวกันอีก 2 ตาราง เพื่อความสะดวกในการเรียกชื่อตัวแปรให้เป็นระบบเดียวกัน เราอาจเพิ่มเลขลำดับอีกตัวหนึ่งคือ J (J = 1 ถึง 3) โดยให้ J=1 แทนชั้น ป.5 J=2 แทน ป.6 และ J=3 แทน ป.7 เป็นต้น

ดังนั้น N$(I,J) หมายถึง <u>ชื่อ</u> ลำดับที่ I ในชั้น J (I=1 ถึง 5, J=1 ถึง 3) ถ้าให้ I=2, J=1 จะได้ N$(2,1) = "แดง" เป็นต้น

หากจะมีการใช้ตัวแปรแถว ที่ต้นโปรแกรมจะต้องมีคำสั่งระบุ **ขนาด** ไว้ด้วย เช่นในกรณีตามตัวอย่างจะเป็นดังนี้

5 ขนาด N$(5,3)

การใช้ตัวแปรแถวมีประโยชน์มากในการทำโปรแกรมชั้นสูง แต่สำหรับโปรแกรมเบื้องต้นอาจไม่มีความจำเป็นนัก เพราะฉะนั้น ถ้ายังอ่านไม่ค่อยเข้าใจ จะข้ามข้อนี้ไปก่อนก็ได้

4.15 ฟังก์ชั่นของตัวแปรอักขระ

ไทยปกาสิต มีฟังก์ชั่นของตัวแปรอักขระอย่างครบครัน คือ

1. **การผนวก** ขบวนอักขระเข้าด้วยกัน (String Concatenation) ซึ่งใช้คำสั่งในรูป **C$ = A$ + B$**
2. การตัดเอามาเฉพาะส่วนซ้ายของขบวนอักขระ จำนวน n ตัว โดยใช้คำสั่ง **C$ = ซ้าย(A$,n)**
3. การตัดอามาเฉพาะส่วนขวาของขบวนอักขระ จำนวน n ตัว โดยใช้คำสั่ง **C$ = ขวา(A$,n)**
4. การตัดเอามาเฉพาะส่วนกลางของขบวนอักขระ ตั้งแต่ตัวที่ m จำนวน n ตัว โดยใช้คำสั่ง **C$ = กลาง(A$,m,n**

อย่างไรก็ตาม การใช้ฟังก์ชั่นเหล่านี้กับขบวนอักขระภาษาไทย มีปัญหาเล็กน้อย ที่เกิดจากวิธีการเข้ารหัสของตัวอักษรภาษาไทย ที่ใช้ติดต่อภายในระหว่างส่วนต่าง ๆ ของคอมพิวเตอร์ ซึ่งต้องใช้อักษรนำ และอักษรปิดท้าย (CTRL-A) เมื่อเริ่มต้นและเมื่อจบขบวนอักขระภาษาไทยทุกครั้ง ดังนั้นถ้าเราเห็นข้อความบนจอเป็นดังนี้

```
10 A$ = "สวัสดี"
20 B$ = "ครับ"
```

ข้อความที่อยู่ในหน่วยความจำจะเป็นดังนี้

```
10 A$ = "(CTRL-A)สวัสดี(CTRL-A)"
20 B$ = "(CTRL-A)ครับ(CTRL-A)"
```

ถ้าเราจะผนวก A$ และ B$ เข้าด้วยกัน และเรียกว่า C$ จะต้องเขียนคำสั่งดังนี้

```
30 A1$ = ซ้าย(A$,7)
40 B1$ = ขวา(B$,5)
50 C$ = A1$ + B1$
```

บรรทัด 30 ตัดเอาส่วนซ้ายของ A$ คือตัด (CTRL-A) ตัวหลังทิ้ง
บรรทัด 40 ตัดเอาส่วนขวาของ B$ คือตัด (CTRL-A) ตัวหน้าทิ้ง
เมื่อมาผนวกกันในบรรทัด 50 จึงมี (CTRL-A) นำหน้า และปิดท้ายถูกต้อง

จงพิมพ์คำสั่ง **ใหม่** แล้วพิมพ์บรรทัด 10 ถึง 50 และลองสั่ง **ทำ** ดู

4.16 การจัดรูปแบบบนจอ

บางครั้งเราต้องการจัดรูปแบบของข้อความบนจอ เช่นจัดให้เป็นตาราง ข้อมูลเป็นต้น ซึ่งเราสามารถจะจัดเป็น 3 แถว (คอลัมน์) ได้โดยใช้เครื่องหมาย , คั่นระหว่างสิ่งที่เราสั่งให้พิมพ์ นอกจากนี้ เรายังสามารถจัดได้โดยใช้คำสั่งที่จะกล่าวในข้อนี้

เว้นวรรค (n) เป็นคำสั่งให้พิมพ์ช่องว่าง จำนวน n ช่อง เช่น

```
100 พิมพ์ "มกราคม";เว้นวรรค(5);"กุมภาพันธ์";เว้นวรรค(5);
    "มีนาคม"
```

⊳ ทำ 100

เลื่อน (n) เป็นคำสั่งให้เลื่อนไปพิมพ์ที่คอลัมน์ n โดยที่ตำแหน่งซ้ายสุดของจอตรงกับคอลัมน์ 1 และตำแหน่งขวาสุดตรงกับคอลัมน์ 40 ดังนั้น n จึงต้องมีค่าในระหว่าง 1 ถึง 40 ตัวอย่างเช่น

```
200 พิมพ์ "12345";เลื่อน(8);"89012";เลื่อน(15);"56789"
```

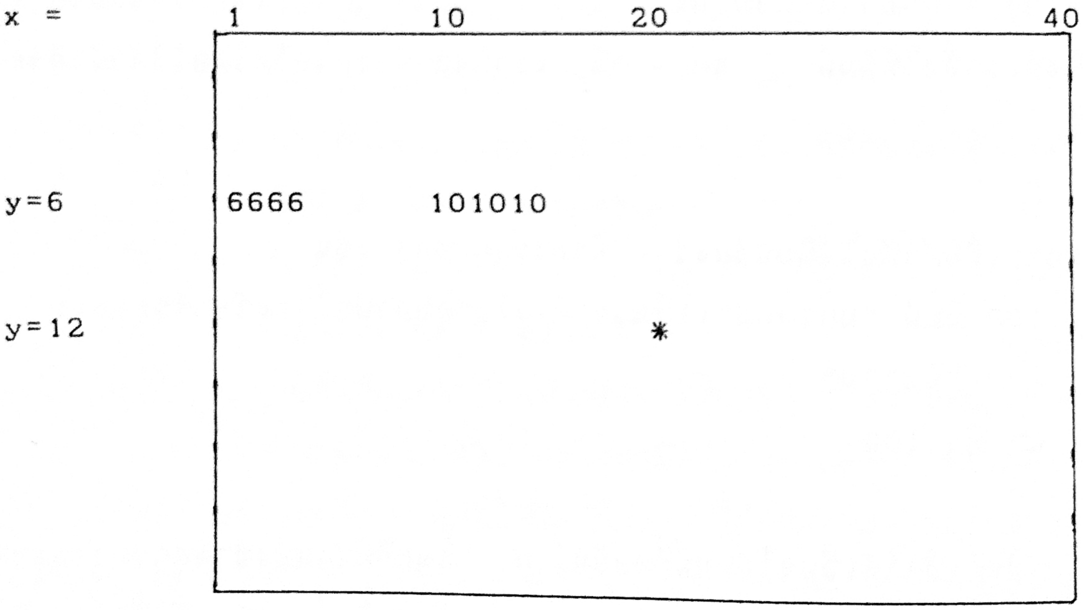
⊳ ทำ 200

ย้ายดิ่ง x และ **ย้ายราบ** y เป็นคำสั่งให้ย้ายเคอร์เซอร์ไปที่ตำแหน่ง (x,y) บน

จอดังแสดงในรูป 4.3 คำสั่งคู่นี้ สามารถแยกใช้เพียงตัวเดียวก็ได้ เช่น ถ้าสั่ง **ย้ายราบ** y ตำแหน่งเคอร์เซอร์จะเปลี่ยนเฉพาะแนวราบเท่านั้น โดยยังรักษาระดับในแนวดิ่งไว้คงเดิม ให้สังเกตว่าคำสั่งคู่นี้ไม่ต้องตามหลังคำสั่ง **พิมพ์** ซึ่งต่างกับ **เว้นวรรค** และ **เลื่อน**

ตัวอย่างเช่น

```
300 ล้างจอ
310 ย้ายดิ่ง 6
320 พิมพ์ "6666";
330 ย้ายราบ 10
340 พิมพ์ "101010"
350 ย้ายดิ่ง 12 : ย้ายราบ 20 : พิมพ์ "*"
▷ ทำ 300
```



รูป 4.3 แสดงการใช้คำสั่ง <u>ย้ายดิ่ง</u> - <u>ย้ายราบ</u>

4.17 สรุป

ผู้อ่านที่ติดตามมาถึงข้อนี้ นับว่ามีความสามารถในการเขียนโปรแกรมในระดับเบื้องต้นได้แล้ว หากจะศึกษาทำความเข้าใจเกี่ยวกับคำสั่งต่าง ๆ เพิ่มเติม และเพิ่มพูนประสบการณ์ขึ้นไปอีก ก็มีโอกาสจะเขียนโปรแกรมระดับสูงได้

ในบทนี้ เราได้เว้นที่จะกล่าวถึงคำสั่งบางคำสั่ง ซึ่งยากเกินไป หรือไม่เหมาะสมที่จะนำมาศึกษาการเขียนโปรแกรมเบื้องต้น

คำสั่งทุกคำสั่งของ **ไทยปกาสิต** ได้รวบรวมไว้อธิบายอย่างละเอียดในบทสุดท้าย โดยเรียงตามลำดับอักษร บทสุดท้ายนี้จึงเป็นเสมือนพจนานุกรมคำสั่งของ **ไทยปกาสิต** และเป็นคู่มืออ้างอิงสำหรับผู้เขียนโปรแกรม

บทที่ 5
ประมวลคำสั่งของไทยปกาสิต

ในบทนี้ จะรวบรวมบรรดาคำสั่งของ **ไทยปกาสิต** มาอธิบายวิธีใช้ ตลอดจนข้อกำหนด และรายละเอียดปลีกย่อยของแต่ละคำสั่ง โดยเรียงตามลำดับอักษร บทนี้จะมีประโยชน์มากสำหรับผู้เขียนโปรแกรมใช้เป็นคู่มืออ้างอิงได้

5.1 ข้อตกลงเกี่ยวกับสัญลักษณ์

สัญลักษณ์ที่ใช้ในบทนี้ มีความหมายจำเพาะที่ผู้อ่านจะต้องทำความเข้าใจเสียก่อน เพื่อมิให้เกิดความสับสน ดังนี้

1. **อักษรภาษาอังกฤษตัวใหญ่ เครื่องหมายวงเล็บ ( ) และคอมมา,** เป็นตัวอักษรและเครื่องหมายที่กำหนดให้ผู้ใช้คำสั่งพิมพ์ตามนั้น
2. **อักษรภาษาอังกฤษตัวเล็ก** แทนค่าตัวเลข, ตัวแปร หรือนิพจน์ที่ผู้ใช้คำสั่งต้องบรรจุลงไป ตามรายละเอียดที่อธิบายในแต่ละคำสั่ง
3. **วงเล็บสี่เหลี่ยม** [ ] ใช้แสดงว่าข้อความในวงเล็บสี่เหลี่ยมนั้นจะมีหรือไม่ก็ได้ ขึ้นอยู่กับการเลือกใช้ของผู้ใช้คำสั่ง
4. **เครื่องหมาย ...** แทน คำสั่ง ข้อความ ตัวแปร ฯลฯ ที่ละไว้

5.2 คำสั่งทำการ

ในหัวข้อนี้ จะกล่าวถึงคำสั่งที่สั่งให้คอมพิวเตอร์ทำงาน ส่วนคำสั่งอีกประเภทหนึ่งคือ **ฟังก์ชั่น** จะได้กล่าวถึงในหัวข้อ 5.3

กระพริบ (FLASH)

สั่งให้แสดงผลแบบกระพริบบนจอ ใช้ได้เฉพาะการแสดงผลด้วยภาษาอังกฤษล้วน

วิธีเขียนคำสั่ง กระพริบ

ตัวอักษรทุกตัวที่ถูกส่งออกไปแสดงบนจอหลังจากคำสั่งนี้ จะสลับไปมาระหว่างภาพปกติ กับ ภาพสลับแสง (กลับขาวเป็นดำ ดำเป็นขาว) ยกเว้นข้อความรายงานความผิดพลาด และตัวอักษรที่รับเข้ามาโดย คำสั่ง **รับ**

หลักการทำงานของคำสั่ง **กระพริบ** อาศัยการเปลี่ยนแปลงรหัสของตัวอักษรที่จะนำออกไปแสดงบนจอ หากตัวอักษรนี้ถูกส่งไปเก็บในดิสค์ เวลาอ่านกลับเข้ามาจะเป็นรหัสที่ไม่ถูกต้อง

กลับ (RETURN) ดู **ไปภาค (GOSUB)**

กำหนดหลักสูตร (DEF FN)

คำสั่งนี้มีไว้ให้ผู้ใช้เครื่องสามารถกำหนดสูตรขึ้นเองได้ โดยสามารถนำสูตรที่กำหนดขึ้นมาใช้ในโปรแกรมภาษาไทยปกาสิตได้

วิธีเขียนคำสั่ง กำหนดสูตร f(x) = e

f เป็นชื่อสูตร ขึ้นต้นด้วยอักษรภาษาอังกฤษตัวใหญ่

x เป็นชื่อตัวแปร ขึ้นต้นด้วยอักษรภาษาอังกฤษตัวใหญ่ ตัวแปรนี้มักจะปรากฏอยู่ในสูตร e ด้วย แต่จะไม่ปรากฏก็ได้

e เป็นสูตรคณิตศาสตร์ใดๆ ที่อาจประกอบด้วยตัวแปร x และตัวแปรอื่น ๆ ที่มีใช้อยู่ในโปรแกรม

การกำหนดสูตรโดยคำสั่งนี้จะต้องทำให้เสร็จในบรรทัดเดียว ถ้ามีการกำหนดสูตรอื่นไว้ก่อนหน้าแล้ว ก็สามารถนำเอาสูตรนี้มาใช้ในการกำหนดครั้งใหม่นี้ได้

คำสั่งนี้ไม่ใช่ **คำสั่งตรง** ไม่สามารถใช้สั่งปฏิบัติทันที

เก็บ (STORE)

สั่งให้บันทึกค่าตัวแปรแถวลงในตลับเทป

วิธีเขียนคำสั่ง เก็บ x

x เป็นชื่อตัวแปร

ในการ **เก็บ** นี้จะต้องกดเครื่องเล่นเทปตลับ ให้ทำงานในหน้าที่อัด (RECORD) รอไว้ก่อน เมื่อเริ่มเก็บ คอมพิวเตอร์จะส่งเสียงตู๊ด 1 ครั้ง และส่งเสียงอีกครั้งหนึ่งเมื่อเก็บเสร็จ

คำสั่งนี้ใช้กับตัวแปรแถวเลขเท่านั้น ถ้าจะใช้กับตัวแปรแถวขบวนอักขระจะต้องเปลี่ยนตัวอักษรแต่ละตัวเป็นตัวเลขก่อน โดยใช้คำสั่ง **รหัส** (ASC)

คำสั่งนี้ใช้คู่กับคำสั่ง **เรียกกลับ** (RECALL)

ขนาด (DIM)

สั่งให้คอมพิวเตอร์สำรองหน่วยความจำไว้เก็บค่าของ **ตัวแปรแถว**

วิธีเขียนคำสั่ง

ขนาด a1(i1) [, a2(i2)][, a3(i3)]......

ขนาด b1(i1,j1)[, b2(i2,j2)][, b3(i3, j3)].....

ขนาด c1(i1, j1, k1)[, c2(i2, j2, k2)][, c3(i3, j3, k3)]...

a1, a2, a3 ฯลฯ เป็นชื่อตัวแปรแถว 1 มิติ

b1, b2, b3 ฯลฯ เป็นชื่อตัวแปรแถว 2 มิติ

c1, c2, c3 ฯลฯ เป็นชื่อตัวแปรแถว 3 มิติ

i1, i2, i3.... เป็นขนาดของตัวแปรมิติที่ 1

ji, j2, j3.... เป็นขนาดของตัวแปรมิติที่ 2

k1, k2, k3.... เป็นขนาดของตัวแปรมิติที่ 3

ตัวอย่าง เช่น **ขนาด** VOL (2,3,4) (สมมติ VOL = x*y*z)

เป็นการกำหนดว่า ตัวแปรชื่อ VOL มี 2 ค่าในมิติที่ 1 (สมมติเป็นค่า X)

มี 3 ค่าในมิติที่ 2 (สมมติเป็นค่า y)

มี 4 ค่าในมิติที่ 3 (สมมติเป็นค่า z)

ในกรณีนี้ เราจะมีค่าของ VOL รวม 24 ค่า เป็นต้น

ตัวแปรแถวอาจเป็นตัวแปรแบบใดแบบหนึ่งใน 3 แบบ คือ เลขจำนวนเต็ม เลขจำนวนจริง หรือขบวนอักขระ ทั้งนี้โดยระบุในชื่อของตัวแปรแถวนั้น เช่น

A% แทนจำนวนเต็ม

B แทนจำนวนจริง

C$ แทนขบวนอักขระ เป็นต้น

จำนวนตัวเลขในวงเล็บที่ตามหลังตัวแปร บอกจำนวนมิติของตัวแปร และค่าตัวเลขเหล่านั้นแทนขนาดหรือจำนวนค่าของตัวแปรในแต่ละมิติ ขนาดของแต่ละมิติถูกจำกัดด้วยหน่วยความจำที่มีเหลืออยู่ขณะนั้น แต่ใหญ่ที่สุดต้องไม่เกิน 88 การระบุขนาดที่ใหญ่เกินไป จะทำให้เครื่องรายงานความผิดพลาดว่า **หน่วยความจำไม่พอ**

หากมีการอ้างถึง **มิติ** ที่สูงกว่าที่ได้กำหนดในคำสั่ง **ขนาด** เครื่อง

จะรายงานความผิดว่า **เลขลำดับตัวแปรแถวผิด**

ถ้ามีการอ้างถึงตัวแปรแถวใด โดยมิได้กำหนด **ขนาด** ไว้ล่วงหน้า คอมพิวเตอร์จะกำหนดขนาดให้เท่ากับ 10 ในแต่ละมิติ การกำหนด**ขนาด** ให้กำหนดได้เพียงครั้งเดียว ในแต่ละโปรแกรม มิฉะนั้น เครื่องจะรายงานความผิดพลาดว่า **กำหนกขนาดซ้ำ**

ข้อมูล (DATA)

เป็นคำสั่งที่ใช้นำหน้าข้อมูลที่เขียนเรียบเรียงไว้ในบรรทัดนั้น เพื่อให้คอมพิวเตอร์อ่านเข้าไป เมื่อพบคำสั่ง **อ่าน**

วิธีเขียนคำสั่ง **ข้อมูล** d1 [,d2,...dn]

d1 (ถึง dn) เป็นค่าคงที่ (ตัวเลข หรือขบวนอักขระ)

คำสั่ง **ข้อมูล** นี้ จะวางไว้ในส่วนใดของโปรแกรมก็ได้ ไม่จำเป็นต้องให้คอมพิวเตอร์รับรู้ก่อนที่ใช้คำสั่ง **อ่าน** คำสั่ง **ข้อมูล** ระบุค่าคงที่ ซึ่งอาจเป็นตัวเลขหรือขบวนอักขระก็ได้ ขบวนอักขระ มักจะอยู่ในเครื่องหมายอัญญประกาศ แต่ที่จริงแล้วไม่จำเป็นต้องมีเครื่องหมายอัญญประกาศก็ได้ เว้นแต่ว่าขบวนอักขระนั้นจะประกอบด้วย **เว้นวรรค คอมม่า (,) หรือ โคลอน (:)** ตัวเครื่องหมายอัญญประกาศเองจะต้องปรากฏเป็นส่วนหนึ่งของขบวนอักขระ เว้นแต่จะเรียกในรูป CHR$(34)

ค่าคงที่บางตัวอาจไม่มีค่าก็ได้ คือเป็นเลข 0 หรือเป็นขบวนอักขระว่างเปล่า ("")

การใช้คำสั่ง **ข้อมูล** เป็นคำสั่งตรง จะไม่มีการรายงานความผิดพลาด แต่ข้อมูลที่ระบุโดยวิธีนี้จะไม่ถูกอ่าน โดยคำสั่ง **อ่าน**

ขอบจำบน (HIMEM)

ใช้กำหนดค่าแอดเดรสสูงสุดในหน่วยความจำ ที่จะยอมให้ใช้เก็บโปรแกรมในภาษาไทยปกาสิต หรือ ภาษาเบสิค รวมทั้งตัวแปร

วิธีเขียนคำสั่ง ขอบจำบน X

X เป็นค่าตัวเลข หรือ นิพจน์ซึ่งมีค่าตัวเลข ระหว่าง - 65535 ถึง 65535

ตามปรกติ เมื่อเริ่มเปิดเครื่องคอมพิวเตอร์ จะกำหนดค่าขอบจำบน

ไว้ที่แอดเดรสสูงสุดของหน่วยความจำ เช่น ที่แอดเดรส 49151 สำหรับหน่วยความจำขนาด 48 k และเมื่อบรรจุ DOS เข้ามา DOS จะครอบครองส่วนบนของหน่วยความจำ รวมประมาณ 10,800 ไบท์ ค่าขอบจำบนก็จะลดลงมาโดยอัตโนมัติ เพื่อไม่ให้โปรแกรมล้ำเข้าไปใน DOS ถ้ามีการใช้คำสั่ง **แฟ้มมากสุด** เครื่องจะสำรองหน่วยความจำอีกแฟ้มละ 595 ไบท์ ถัดลงมา อย่ากำหนดค่า **ขอบจำบน** สูงกว่าแอดเดรสสูงสุดของหน่วยความจำทีมีอยู่ มิฉะนั้นค่าตัวแปรบางตัวอาจสูญหายได้

ถ้าต้องการทราบค่า ขอบจำบน ณ ขณะนั้น ให้ใช้คำสั่งต่อไปนี้

พิมพ์ หยิบ (116) * 256 + หยิบ (115)

ขอบจำบน จะไม่เปลี่ยนแปลง ภายหลังการใช้คำสั่ง **ใหม่ ทำ** และ **ล้าง**

ถ้ากำหนดว่า **ขอบจำบน** ไว้ต่ำกว่า **ขอบจำล่าง** หรือกำหนด **ขอบจำบน** ไว้ต่ำเกินไปจนไม่มีที่พอสำหรับเก็บโปรแกรม เครื่องจะรายงานความผิดพลาดออกมาว่า **หน่วยความจำไม่พอ**

ขอบจำล่าง (LOMEM)

ใช้กำหนดค่าแอดเดรสต่ำสุดในหน่วยความจำที่จะยอมให้ใช้กับโปรแกรมในภาษาไทยปกาสิต หรือ ภาษาเบสิครวมทั้งตัวแปร

วิธีเขียนคำสั่ง ขอบจำล่าง x

x เป็นค่าตัวเลข หรือ นิพจน์ซึ่งมีค่าตัวเลข

ระหว่าง -65535 ถึง 65535

ตามปรกติ คอมพิวเตอร์จะกำหนดค่า **ขอบจำล่าง** เองโดยอัตโนมัติอยู่แล้ว เพื่อกันหน่วยความจำส่วนที่อยู่ต่ำกว่าขอบจำล่างนี้ไว้ใช้งานเอง ในกรณีที่ผู้ใช้เครื่องต้องการป้องกันโปรแกรมบางส่วนของตน เช่นโปรแกรมย่อยภาษาเครื่องหรือตารางรูปร่างของภาพละเอียด ก็สามารถใช้คำสั่งนี้กันเนื้อที่หน่วยความจำส่วนล่างเพิ่มเติมได้ ถ้าต้องการทราบค่า ขอบจำล่าง ณ ขณะนั้น ให้ใช้คำสั่ง

พิมพ์ หยิบ (106)* 256 + หยิบ (105)

ถ้ากำหนดค่า **ขอบจำล่าง** ไว้สูงกว่า **ขอบจำบน** หรือกำหนดไว้ต่ำกว่าค่า **ขอบจำล่าง** ที่คอมพิวเตอร์จัดให้ หรือต่ำกว่าแอดเดรสสูงสุดที่ DOS ใช้อยู่ เครื่องจะรายงานความผิดพลาดว่า **หน่วยความจำไม่พอ**

104
น.จ พ.5487

001.6424
ย136 ค

ขึ้นกับ...ไป (ON...GOTO)

สั่งให้คอมพิวเตอร์เลือกกระโดดไปทำยังเลขบรรทัดที่ระบุไว้หลายแห่ง

วิธีเขียนคำสั่ง ขึ้นกับ e ไป n1,[n2,n3,...,nn]

e เป็นตัวแปรเลข, สมาชิกของตัวแปรแถวหรือนิพจน์ใด ๆ

n1 ถึง nn เลขบรรทัดของคำสั่งในโปรแกรม

โปรแกรมจะกระโดดไปทำบรรทัด n1 ถ้า ค่าจำนวนเต็มของ e เป็น 1 และจะกระโดดไปทำบรรทัด n2 ถ้า ค่าจำนวนเต็มของ e เป็น 2 ฯลฯ

ค่าของ e จะต้องอยู่ระหว่าง 0 ถึง 255 มิฉะนั้นเครื่องจะรายงานความผิดพลาดว่า **ค่าเกินขอบเขต** ถ้า e มีค่าเป็น 0 หรือมากกว่าจำนวนของตัวเลขบรรทัดในคำสั่งนี้ โปรแกรมจะทำคำสั่งที่ถัดไปจากคำสั่ง ขึ้นกับ...ไป

ขึ้นกับ...ไปภาค (ON GOSUB)

สั่งให้คอมพิวเตอร์กระโดดออกจากโปรแกรมหลัก ไปเลือกทำโปรแกรมย่อย (ภาค) ที่ระบุไว้หลายแห่งขึ้นอยู่กับค่าของ e

วิธีเขียนคำสั่ง ขึ้นกับ e ไปภาค n1,[,n2,n3,...,nk]

e เป็นตัวแปรเลข, สมาชิกของตัวแปรแถวหรือนิพจน์ใดๆ

n1 ถึง nk เป็นเลขบรรทัดของคำสั่งในโปรแกรม

โปรแกรมจะกระโดดไปทำโปรแกรมย่อยซึ่งเริ่มที่บรรทัด n1 ถ้า e มีค่าจำนวนเต็ม 1 และไปที่ n2 ถ้า e มีค่าจำนวนเต็ม เป็น 2 เป็นต้น เมื่อทำโปรแกรมย่อย ภาคใดภาคหนึ่งจนถึงคำสั่ง **กลับ** แล้วจะกลับมาที่โปรแกรมหลักและจะทำบรรทัดที่ถัดจากคำสั่ง **ขึ้นกับ...ไปภาค** ต่อไป

ความเร็ว (SPEED)

สั่งให้เปลี่ยนอัตราเร็วในการแสดงตัวอักษรบนจอภาพ

วิธีเขียนคำสั่ง ความเร็ว e

e เป็นตัวเลขจำนวนเต็มระหว่าง 0 ถึง 255 หรือนิพจน์ที่ให้ค่าตัวเลขเช่นนี้

ความเร็ว 0 จะให้ความเร็วต่ำสุด ส่วนความเร็ว 255 จะให้ความเร็วสูงสุด

จบ **(END)**

สั่งให้หยุดการทำงานตามโปรแกรม

วิธีเขียนคำสั่ง จบ

ในการใช้ไทยปกาสิต คำสั่ง **จบ** ไม่ต้องใช้ก็ได้ เพราะเมื่อโปรแกรมทำมาถึงบรรทัดสุดท้ายแล้ว จะจบโดยอัตโนมัติ แต่คำสั่ง **จบ** มักจะใช้เขียนไว้ในบรรทัดสุดท้ายของโปรแกรมหลัก และให้โปรแกรมย่อย (ภาค) ต่างๆ ต่อหลังบรรทัดนั้น เป็นการแยกโปรแกรมหลักกับโปรแกรมย่อยไม่ให้ทำต่อเนื่องกัน

หลังคำสั่ง **จบ** คอมพิวเตอร์จะแสดง **พร้อมท์** (PROMPT) ซึ่งแสดงว่าออกจากโปรแกรมแล้ว และไทยปกาสิตรอรับคำสั่ง

จอปกติ **(TEXT)**

คำว่าจอปกติ ในที่นี้หมายถึงจอภาพที่แสดงตัวอักษร หรือข้อความเต็มจอ คำสั่ง **จอปกติ** นี้ ใช้สั่งให้การแสดงภาพบนจอ กลับคืนเข้าสู่สภาพจอปกติ หลังจากที่เคยถูกสั่งให้แสดงภาพหยาบ (ด้วยคำสั่ง **ภย**) หรือภาพละเอียด (ด้วยคำสั่ง **ภล** หรือ **ภลต**)

วิธีเขียนคำสั่ง จอปกติ

ผลจากการปฏิบัติตามคำสั่งนี้คือ เครื่องหมาย **พร้อมท์** และ **เคอร์เซอร์** จะไปปรากฏที่บรรทัดสุดท้ายบนจอ คำสั่งนี้ไม่ล้างหน่วยความจำ หน้า 1 ของภาพหยาบ (PAGE-1 LORES GRAPHIC) ดังนั้น ถ้าสั่ง **จอปกติ** หลังจากการแสดงภาพหยาบแล้ว ในส่วนบนของจอ (20 บรรทัดแรก) อาจมีตัวอักษรประหลาดปรากฏขึ้นได้

จับ **(GET)**

สั่งให้คอยตรวจดูว่ามีตัวอักษร หรือตัวเลขตัวใดถูกพิมพ์เข้ามาทางแป้นพิมพ์ และให้นำตัวอักษรหรือตัวเลขนั้น ไปกำหนดเป็นค่าของตัวแปร โดยไม่แสดงบนจอภาพ

วิธีเขียนคำสั่ง จับ x หรือ จับ x$

x = ตัวแปรเลข, x$ = ตัวแปรอักขระ

เมื่อคอมพิวเตอร์ได้รับคำสั่ง **จับ** จะหยุดรอจนกว่าแป้นพิมพ์ถูกกด ถ้า V เป็นตัวแปรอักขระ อักษรที่รับเข้ามาจะถูกกำหนดเป็นค่าของตัวแปร ถ้ารับ CTRL-@ เข้ามา **ขบวนอักขระว่างเปล่า** จะถูกกำหนดเป็นค่าของตัวแปร

คำสั่ง **จับ** มักไม่ใช้กับตัวแปรเลข แต่ก็สามารถใช้ได้ ในกรณีนี้ ถ้ากดเลข 0 ถึง 9 จะกำหนดค่าตัวเลขนี้ให้แก่ตัวแปร การกดเครื่องหมายต่อไปนี้ จะให้ค่า 0 แก่ตัวแปร คือ CTRL-@, เว้นวรรค, E, และจุดทศนิยม (.) การกดแป้นอักษรอื่นใด นอกจากแป้นอักษรที่กล่าวแล้ว เครื่องจะรายงานความผิดพลาดว่า **ไวยากรณ์ผิด** และโปรแกรมจะหยุดทำงาน

คำสั่ง **จับ** ไม่สามารถใช้เป็น **คำสั่งตรง** ได้

จาก...ถึง (FOR TO)

คำสั่งนี้ใช้คู่กับ **เพิ่ม** (NEXT) เป็นคำสั่งที่สั่งให้คอมพิวเตอร์วนทำซ้ำในส่วนของโปรแกรมที่กำหนด เป็นจำนวนรอบที่ต้องการ

วิธีเขียนคำสั่ง จาก v = i ถึง f [ทีละ s]

....

เพิ่ม [v]

v เป็นตัวแปรควบคุม

i เป็นค่าเริ่มต้นของ v (อาจเป็นนิพจน์ได้)

f เป็นค่าสุดท้ายของ v (อาจเป็นนิพจน์ได้)

s เป็นค่าส่วนเพิ่มแต่ละขั้นของ v (อาจเป็นนิพจน์ได้)

เมื่อเริ่มคำสั่ง **จาก** v จะถูกกำหนดค่าให้เท่ากับ i จากนั้น โปรแกรมส่วนที่อยู่ถัดมาจะทำ 1 ครั้ง จนถึงคำสั่ง **เพิ่ม** ค่าของ v จะถูกบวกด้วย s ถ้าผลบวก (ค่าใหม่ของ v) ยังไม่เกิน f โปรแกรมจะย้อนกลับไปทำบรรทัดที่ถัดลงมาจากคำสั่ง **จาก** อีกครั้งหนึ่ง และวนเช่นนี้เรื่อยไป จนกว่าค่าใหม่ของ v จะมีค่าเกิน f

s มักจะเป็นค่าบวก แต่จะใช้ค่าลบก็ได้ ในกรณีนี้ต้องให้ f มีค่าน้อยกว่า i การทำงานของโปรแกรมจะวนจนกว่า v จะมีค่าน้อยกว่า f

ถ้าส่วนที่ระบุ **ทีละ** s ไม่มี เครื่องจะถือว่า s = +1

คำสั่ง จาก...เพิ่ม สามารถจะเขียนซ้อนกันได้ถึง 10 ชั้น แต่ต้องใช้

ตัวแปรชื่อต่างกัน

จุด (PLOT)

แสดงจุด 1 จุด ณ ตำแหน่งที่ระบุบนจอในลักษณะภาพหยาบ

วิธีเขียนคำสั่ง จุด c, r

c เป็นค่าคอลัมน์ (ตำแหน่งช่องอักษร) ซึ่งมีค่า 0-39 คอลัมน์ 0 อยู่ที่ขอบจอด้านซ้าย คอลัมน์ 39 อยู่ด้านขวา

r เป็นค่าระดับ (ตำแหน่งบรรทัดหรือช่องว่างระหว่างบรรทัด) ซึ่งมีค่าต่อ 0-47 โดยที่ 0 อยู่ด้านบน และ 47 อยู่ด้านล่างของจอ ระดับ 40 ถึง 47 ตามปกติจะสำรองไว้สำหรับพิมพ์ข้อความ 4 บรรทัด หากต้องการใช้แสดงภาพหยาบในบริเวณนี้ ต้องใช้คำสั่ง **หยอด** - 16302, 0 เพื่อบอกเลิกการสำรองบรรทัดข้อความดังกล่าว

หมายเหตุ 1. คำสั่ง **จุด** นี้เป็นคำสั่งที่ใช้สร้างภาพหยาบ จึงต้องตามหลังคำสั่ง **ภย** ซึ่งเปลี่ยนสภาพจอจาก **จอปกติ** (สำหรับแสดงข้อความ) ไปเป็น **จอภาพหยาบ** (สำหรับแสดงภาพหยาบ) หากเราใช้คำสั่ง **จุด** ในขณะที่จอยังเป็นจอปกติอยู่นั้น แทนที่จะได้จุด เราจะได้ตัวอักษร 1 ตัวแทน

2. สีของจุด (หากใช้จอสี) สามารถกำหนดได้ด้วยคำสั่ง **สีจุด** ซึ่งจะต้องมาก่อนคำสั่ง **จุด**

แจง (LIST)

สั่งให้คอมพิวเตอร์นำโปรแกรมที่เก็บไว้ในหน่วยความจำ ทั้งหมดหรือบางส่วนออกมาแสดงบนจอ โดยจัดเรียงตามลำดับเลขบรรทัด

วิธีเขียนคำสั่ง

(ก) แจง (แสดงทั้งโปรแกรม)

(ข) แจง n (แสดงเฉพาะบรรทัด n)

(ค) แจง n1, n2 (แสดงตั้งแต่บรรทัด n1 ถึงบรรทัด n2)

(ง) แจง n1, (แสดงตั้งแต่บรรทัด n1 จนจบ)

(จ) แจง , n2 (แสดงตั้งแต่ต้นถึงบรรทัด n2)

หมายเหตุ เครื่องหมาย , ในกรณี (ค) (ง) และ (จ) อาจใช้เครื่องหมาย - แทนได้

ช่อง (PR#)

สั่งคอมพิวเตอร์ให้เลือกช่องเสียบอุปกรณ์ เพื่อส่งข้อมูลออกไปแสดงผลทางช่องนั้น

วิธีเขียนคำสั่ง ช่อง s

s เป็นหมายเลขช่องเสียบอุปกรณ์ (สลอต) ซึ่งมีค่าได้ตั้งแต่ 0 ถึง 7

คำสั่ง ช่อง ที่ใช้ตามปกติมีดังนี้

คำสั่ง	**ความหมายของคำสั่ง**	**เงื่อนไข**
ช่อง 0	แสดงผลทางจอภาพแบบ 40 คอลัมน์	สั่งเมื่อต้องเลิกใช้ช่องอื่นที่ใช้อยู่ก่อน
ช่อง 1	แสดงผลทางเครื่องพิมพ์	ต้องมี printer-interface card เสียบอยู่ที่ช่อง 1
ช่อง 3	แสดงผลทางจอภาพแบบ 80 คอลัมน์	ต้องมี 80 -column card เสียบอยู่ที่ช่อง 3
ช่อง 6	ให้ติดต่อกับดิสก์ไดร์ฟ	ต้องมี disk-controller card เสียบอยู่ที่ช่อง 6

หมายเหตุ 1. ถ้าไม่มี card เสียบอยู่ตามช่องที่ต้องมีเมื่อใช้คำสั่งช่องนั้นๆ คอมพิวเตอร์จะชงักงัน (อาการเช่นนี้มักเรียกกันว่า แฮ็ง = hang) ต้องทำการ reset จึงจะทำงานต่อได้

2. ในกรณีที่ DOS อยู่ในหน่วยความจำ คำสั่งช่อง จะเป็นคำสั่งของ DOS การใช้คำสั่งนี้ในโปรแกรม จะต้องนำด้วย **พิมพ์** (CTRL-D) หรือ **พิมพ์ รหัส** (4)

เดินต่อ (RESUME)

สั่งให้คอมพิวเตอร์เริ่มทำงานต่อ หลังจากที่หยุดไปเพราะเกิดความผิดพลาด โดยเริ่มจากคำสั่งที่เกิดความผิดพลาดนั้น

วิธีเขียนคำสั่ง เดินต่อ

คำสั่งนี้ควรใช้ในกรณีที่มีการใช้คำสั่ง **เมื่อผิด ไป** ซึ่งมีการแก้ไขข้อผิดพลาดแล้วเท่านั้นจึงจะสั่ง **เดินต่อ** มิฉะนั้นอาจเกิดผลเสียหายที่ทำนายไม่ได้

คำสั่งนี้ไม่ใช่คำสั่งตรง

โดด (POP)

สั่งให้คอมพิวเตอร์ลบตำแหน่ง **กลับ** ของคำสั่ง **ไปภาค** ครั้งล่าสุดออกจากหน่วยความจำ

วิธีเขียนคำสั่ง โดด

คำสั่งนี้มีผลเสมือนหนึ่งว่า คำสั่ง **ไปภาค** ครั้งล่าสุดถูกเปลี่ยนเป็นคำสั่ง **ไป** เมื่อโปรแกรมดำเนินมาจนถึงคำสั่ง **กลับ** ก็จะกลับไปยังบรรทัดถัดจากคำสั่ง **ไปภาค** รองล่าสุด หากจำนวนครั้งที่ทำตามคำสั่ง **โดด** และ **กลับ** รวมแล้วมากกว่าจำนวนครั้งที่ทำตามคำสั่ง **ไปภาค** เครื่องจะรายงานความผิดพลาด

คำสั่งโดดนี้ เป็นคำสั่งที่ใช้แก้ปัญหาเฉพาะหน้า ในกรณีที่มีการใช้คำสั่ง **ไป** เข้าไปทำให้ส่วนใดส่วนหนึ่งของโปรแกรมย่อย (ภาค) ซึ่งวิธีการเช่นนี้ถือว่าเป็นวิธีที่ไม่ดีไม่สมควรถือเป็นแบบอย่าง

ตัด (DEL)

ตัดบรรทัดคำสั่งตามที่ระบุ

วิธีเขียนคำสั่ง ตัด n1,n2

n1,n2 เป็นเลขบรรทัด (n1>n2)

คำสั่งทุกบรรทัดในระหว่าง n1 และ n2 จะถูกลบจากหน่วยความจำ ถ้าไม่ระบุ n1 การตัดจะเริ่มจากบรรทัดถัดจากคำสั่งนี้ ถ้าไม่ระบุ n2 การตัดจะตัดจากบรรทัดแรกมาจนถึงบรรทัดก่อนคำสั่งนี้ ถ้า n1=n2 จะตัดบรรทัดเดียว

ถ้าใช้คำสั่งในโปรแกรม จะมีการตัดบรรทัดตามสั่ง แต่หลังจากนั้นการทำงานนั้นจะหยุดโดยไม่สามารถใช้คำสั่ง **ทำต่อ** ได้

ตาม (TRACE)

แสดงเลขบรรทัดของแต่ละคำสั่งบนจอขณะที่ทำตามคำสั่งนั้น

วิธีเขียนคำสั่ง ตาม

คำสั่งนี้เปิดโอกาสให้ผู้ใช้เครื่อง สามารถติดตามขั้นตอนการทำงานของโปรแกรมได้อย่างใกล้ชิด มีประโยชน์ในการค้นหาข้อผิดพลาดในโปรแกรม

(ดู เลิกตามด้วย)

ทำ (RUN)

สั่งให้เริ่มทำงานตามโปรแกรมที่อยู่ในหน่วยความจำ

วิธีเขียนคำสั่ง ทำ หรือ ทำ [n]

n เป็นเลขบรรทัด

ถ้าไม่มีเลขบรรทัด n จะเริ่มทำจากต้นโปรแกรม(เลขบรรทัดน้อยที่สุด) ถ้ามีเลขบรรทัด n จะเริ่มจากบรรทัดนั้น

ถ้า n ระบุเลขบรรทัดที่ไม่มีในโปรแกรม เครื่องจะรายงานความผิดพลาดว่า "คำสั่งไม่ปรากฏ"

ทำต่อ (CONT)

สั่งให้คอมพิวเตอร์ทำงานตามโปรแกรมเดิมต่อไปหลังจากที่ถูกสั่งให้หยุด

วิธีเขียนคำสั่ง ทำต่อ

ทำต่อ จะสั่งให้ทำงานต่อหลังจากที่หยุดไปเพราะคำสั่ง **หยุด** **จบ** หรือ (CTRL-C) ถ้าคำสั่ง **รับ** ถูกชงักด้วย (CTRL-C) โปรแกรมจะทำต่อไม่ได้ กรณีต่อไปนี้ก็ทำไม่ได้เช่นกัน คือ

1. เมื่อไม่มีบรรทัดของโปรแกรมเหลืออยู่
2. ไม่มีการแก้ไขหรือเพิ่มเติมบรรทัดโปรแกรมหลังจากหยุด
3. มีข้อผิดพลาดเกิดขึ้น

ในกรณีที่ทำต่อไม่ได้เครื่องจะรายงานความผิดพลาดว่า **"ทำต่อไม่ได้"**

บรรจุ (LOAD)

สั่งให้บรรจุโปรแกรมลงในหน่วยความจำ โดยอ่านจากตลับเทป

(สำหรับการ **บรรจุ** จากแผ่นดิสค์ ดูในหนังสือคู่มือ DOS)

วิธีเขียนคำสั่ง บรรจุ

บรรจุโปรแกรมแรกที่อ่านพบจากตลับเทป ลงในหน่วยความจำ แทนที่โปรแกรมเดิม ก่อนใช้คำสั่งนี้ต้องกดเครื่องเล่นเทปตลับ ให้ทำงานในหน้าที่ **เล่น** (PLAY) รออยู่ คอมพิวเตอร์จะส่งเสียงดังปิ๊บเมื่อเริ่มต้นบรรจุ และเมื่อสิ้นสุดการบรรจุ หลังจากเสียงดังปิ๊บครั้งที่ 2 แล้ว ก็สามารถหยุดเครื่องเล่นเทปตลับได้

การบรรจุโปรแกรมจากตลับเทป มักพบปัญหายุ่งยากที่เกิดจากการปรับปุ่มความดัง (VOLUME) และปุ่มแต่งเสียง (TONE) ไว้ไม่พอดี (ข้อแนะนำทั่ว ๆ ไปคือ ให้ปรับปุ่มแต่งเสียงให้แหลมสุดและปรับปุ่มความดังไว้ที่ระดับประมาณ 3 ใน 4) จงทดลองหาตำแหน่งที่พอดี แล้วจดบันทึกไว้

บรรจุภาพ (SHLOAD)

สั่งให้บรรจุตารางรูปร่างภาพละเอียดจากตลับเทป

วิธีเขียนคำสั่ง บรรจุภาพ

ตำแหน่งเริ่มต้นของหน่วยความจำสำหรับบรรจุตารางรูปร่างจะถูกเขียนไว้ ณ ตำแหน่ง 22 และ 23 ก่อนใช้คำสั่งบรรจุภาพให้สั่ง **ขอบจำบน** เสียก่อน เพื่อมิให้ไปทับหน่วยความจำที่ใช้งานอย่างอื่น

บันทึก (SAVE)

บันทึกโปรแกรมในหน่วยความจำลงในตลับเทป

(สำหรับการบันทึกลงในแผ่นดิสค์ ดูในหนังสือคู่มือ DOS)

วิธีเขียนคำสั่ง บันทึก

เครื่องเล่นเทปจะถูกกดให้ทำหน้าที่ **บันทึก (RECORD)** รออยู่ก่อนที่จะใช้คำสั่งนี้ คอมพิวเตอร์จะส่งเสียงดังปิ๊บก่อนบันทึกและเมื่อสิ้นสุดการบันทึก ให้หยุดเครื่องเล่นเทปหลังจากเสียงดังครั้งที่ 2

ปกติ (NORMAL)

สั่งให้จอภาพกลับคืนสู่สภาพปกติ หลังจากที่เคยสั่งให้แสดงภาพ **สลับแสง (INVERSE)** หรือ **กระพริบ (FLASH)**

วิธีเขียนคำสั่ง ปกติ

ไป (GOTO)

สั่งให้ไปทำตามคำสั่งส่วนอื่นซึ่งเริ่มจากเลขบรรทัดที่ระบุ

วิธีเขียนคำสั่ง ไป n

n เป็นหมายเลขบรรทัด (ห้ามเป็นนิพจน์)

ถ้าคำสั่งอ้างถึงหมายเลขบรรทัด n ที่ไม่มีอยู่ในโปรแกรม เครื่องจะรายงานความผิดพลาดว่า "คำสั่งไม่ปรากฏ"

ไปภาค (GOSUB)

สั่งให้คอมพิวเตอร์แยกจากโปรแกรมหลัก ไปทำงานในส่วนของโปรแกรมย่อย(ภาค) ซึ่งเริ่มจากเลขบรรทัดที่ระบุ

วิธีเขียนคำสั่ง ไปภาค n

n เป็นเลขบรรทัด (ห้ามเป็นนิพจน์)

คำสั่ง **ไปภาค** นี้ใช้คู่กับคำสั่ง **กลับ** (RETURN) ซึ่งจะส่งการทำงานกลับมาทำต่อในบรรทัดถัดจากคำสั่ง **ไปภาค**

เราอาจใช้ **ไปภาค...กลับ** ซ้อนกันได้ถึง 24 ระดับ ตามปกติควรหลีกเลี่ยงการใช้คำสั่ง **ไป** ที่จะส่งการทำงานกลับออกจากโปรแกรมย่อย(ภาค) แต่ถ้าจำเป็นต้องทำเช่นนั้น ให้นำหน้าคำสั่ง **ไป** ด้วยคำสั่ง **โดด**

พิมพ์ (PRINT)

แสดงอักษรบนจอภาพหรืออุปกรณ์แสดงผลชนิดอื่นๆ

วิธีเขียนคำสั่ง

1. ใช้กับตัวเลข หรือนิพจน์

   พิมพ์ n1 [; n2 ; n3...]

   พิมพ์ e1 [; e2 ; e3...]

   n1, n2, n3 ฯลฯ เป็นตัวเลขหรือตัวแปรเลข

   e1, e2, e3 ฯลฯ เป็นนิพจน์

   เครื่องหมาย ; อาจเปลี่ยนเป็น , ก็ได้ แต่มีผลไม่เหมือนกัน (ดูข้อ 3.4)

ค่าตัวเลขที่เป็นลบจะมีเครื่องหมายลบนำหน้า ส่วนค่าตัวเลขที่เป็นบวกจะไม่มีอะไรนำหน้า ถ้าตัวเลขซึ่งมีขนาดใหญ่เกินไป หรือเล็กเกินไป จะถูกพิมพ์ด้วย

วิธีแสดงตัวเลขแบบวิทยาศาสตร์

2. ใช้กับขบวนอักขระ

พิมพ์ "a1$ [; a2$; a3$ ...]"

พิมพ์ "e1$ [; e2$; e3$ ...]"

a1$, a2$, a3$ เป็นขบวนอักขระ หรือตัวแปรอักขระ

e1$, e2$, e3$ เป็นนิพจน์ของขบวนอักขระ

เครื่องหมาย ; อาจเปลี่ยนเป็น , ก็ได้ แต่มีผลไม่เหมือนกัน (ดูข้อ 3.4)

คำสั่ง **พิมพ์** สามารถใช้เครื่องหมาย ? แทนได้

เพิ่ม (NEXT)

เป็นส่วนหนึ่งของคำสั่ง **จาก...ถึง** ซึ่งบังคับให้โปรแกรมวนกลับไปทำงานเป็นรอบๆ คำสั่ง **เพิ่ม** เป็นคำสั่งที่สั่งให้เพิ่มค่า **ตัวนับรอบ** หลังจากที่ทำมาครบอีกรอบแล้ว ก่อนที่วนกลับไปทำรอบใหม่

วิธีเขียนคำสั่ง เพิ่ม i [; j, k, ...]

i, j, k เป็นชื่อตัวแปรของ ตัวนับรอบ หรือจะเป็นนิพจน์ของตัวแปรนั้น ๆ ก็ได้ ในกรณีที่มีหลายตัว ต้องเรียงลำดับให้ถูกต้อง โดยตัวแปรของคำสั่ง **จาก...ถึง** ชุดหลังสุดจะต้องถูก **เพิ่ม** ก่อนสุดถ้าไม่ระบุชื่อตัวแปรเครื่องจะถือว่าให้เพิ่มค่าตัวแปรของคำสั่ง **จาก...ถึง** ชุดหลังสุด

ภย (GR)

สั่งให้การแสดงผลบนจอภาพเป็นแบบ **ภาพหยาบ** (40 คูณ 40) โดยเว้นด้านล่างไว้สำหรับแสดงตัวอักษร 4 บรรทัด

วิธีเขียนคำสั่ง ภย

ส่วนแสดงภาพจะปรากฏเป็นพื้นสีดำ เคอร์เซอร์จะไปอยู่ในบริเวณ 4 บรรทัดล่างที่ใช้แสดงตัวอักษร

การเข้าสู่ภาวะ **ภย** มีข้อควรทราบดังนี้

1. จาก ภล (ภาพละเอียด) สามารถเข้าสู่ ภย ได้โดยตรง
2. จาก **ภลต** (ภาพละเอียดเต็มจอ) ให้กลับคืนสู่ **จอปกติ** เสียก่อนแล้วจึงเข้า ภย มิฉะนั้นตัวอักษรประหลาดซึ่งอยู่ในหน่วยความจำจะถูกนำแสดงออกบนจอ
3. ถ้าต้องการแสดงภาพเต็มจอในภาวะ **ภย** ให้ใช้คำสั่ง **หยอด - 16302,0** จะได้ภาพขนาด 40 คูณ 48 จากนั้นทุกอย่างที่พิมพ์ทางแป้นพิมพ์จะถูกแสดงเป็นจุดสีในบริเวณ 4 บรรทัดล่าง ถ้าต้องการแสดงตัวอักษรใน 4 บรรทัดล่างตามเดิมให้ใช้คำสั่ง

หยอด - 16301,0

ภล (HGR)

สั่งให้คอมพิวเตอร์เข้าสู่ภาวะการแสดงภาพละเอียด (280 คูณ 160 จุด) โดยเว้นด้านล่างไว้สำหรับแสดงตัวอักษร 4 บรรทัด

วิธีเขียนคำสั่ง ภล

เมื่อใช้คำสั่งนี้ หน่วยความจำภาพละเอียดหน้า 1 จะถูกนำออกแสดงบนจอ ส่วนหน่วยความจำของตัวอักษรจะยังคงเดิม แต่เฉพาะ 4 บรรทัดล่างเท่านั้นจะถูกนำออกแสดงบนจอ เคอร์เซอร์จะยังอยู่ในตำแหน่งเดิม ฉะนั้นอาจต้องพิมพ์ข้อความหลายบรรทัด (หรือกด RETURN หลายครั้ง) ก่อนที่เคอร์เซอร์จะมาปรากฏที่ 4 บรรทัดล่าง ส่วนแสดงภาพจะปรากฏเป็นสีดำ และค่า ภลสี คงเดิม

ถ้าต้องการภาพละเอียดเต็มจอให้ใช้คำสั่ง

หยอด - 16302,0 (หลังจากคำสั่ง ภล)

หลังจากนั้นคำสั่งต่างๆ ที่ป้อนเข้าทางแป้นพิมพ์จะไม่ปรากฏบนจอ แต่จะทำงานได้ตามปกติ ถ้าต้องการแสดงตัวอักษรใน 4 บรรทัดล่างตามเดิมให้ใช้คำสั่ง

หยอด - 16301,0

ในกรณีที่ภาษาเบสิคอยู่ในดิสค์ เมื่อบรรจุภาษาเบสิคเข้าหน่วยความจำจะไปทับส่วนหนึ่งของหน่วยความจำภาพละเอียดหน้า 1 ดังนั้นจึงไม่สามารถใช้ **ภล** ได้กับภาษาเบสิคแบบนี้

ในกรณีที่ใช้ภาษาเบสิคแบบเฟิร์มแวร์ (เช่น ไทยปกาสิต) ถ้าโปรแกรมมีความยาวมากๆ อาจล้นขึ้นไปทับหน้า 1 ของหน่วยความจำภาพละเอียดได้

หากจะป้องกันมิให้โปรแกรมขึ้นไปทับภาพละเอียด ให้ใช้คำสั่ง

ขอบจำบน 8192

ภลจุด (HPLOT)

สั่งให้วางจุดสี หรือลากเส้นสีบนจอภาพละเอียด

วิธีเขียนคำสั่ง

(ก) ภลจุด x, y

(ข) ภลจุด ถึง x, y

(ค) ภลจุด x1, y1 ถึง x2, y2 [ถึง x3, y3...]

(ก) วางจุด ณ ตำแหน่งที่ระบุโดยใช้สีที่สั่งด้วย **ภลสี** ครั้งล่าสุด

(ข) ลากเส้นจากจุดสุดท้ายที่วาง หรือจากปลายเส้นที่วาดล่าสุด ไปยังตำแหน่งที่กำหนด ถ้ายังไม่มีจุดอยู่ก่อนจะไม่มีการลากเส้น สีตามคำสั่ง **ภลสี** ครั้งล่าสุด

(ค) ลากเส้นระหว่าง 2 จุดที่กำหนด และลากต่อไปยังจุดต่อ ๆ ไปได้ด้วยสีตามคำสั่ง **ภลสี** ครั้งล่าสุด

หมายเหตุ

1. ก่อนสั่ง **ภลจุด** ให้สั่ง **ภล** หรือ **ภลต** ก่อนทุกครั้ง มิฉะนั้นอาจมีการทับโปรแกรมหรือค่าตัวแปร
2. เส้นที่ลากในส่วนของ 4 บรรทัดล่างที่ใช้แสดงข้อความ (ในสภาวะ ภล) จะไม่ปรากฏบนจอ

ภลต (HGR 2)

สั่งให้คอมพิวเตอร์เข้าสู่ภาวะ การแสดงภาพละเอียดเต็มจอ (280 คูณ 192 จุด) โดยยำภาพในหน่วยคความจำภาพละเอียด หน้า 2 ออกแสดง

วิธีเขียนคตำสั่ง ภลต

เมื่อใช้คำสั่งนี้ หน่วยความจำของตัวอักษรจะคงเดิม สิ่งที่พิมพ์เข้าไปทางแป้นพิมพ์จะไม่ปรากฏบนจอ แต่จะทำงานได้ตามปรกติ จะมีการล้างจอให้เป็นสีดำ ค่า **ภลสี** จะคงเดิม

ในภาวะนี้จะไม่สามารถแสดงตัวอักษรใน 4 บรรทัดล่างได้ ถ้าใช้คำสั่ง **หยอด** -16301,0 เครื่องจะไม่แสดงตัวอักษรใน 4 บรรทัดล่าง แต่จะแสดง

ภาพหยาบจากหน่วยความจำภาพหยาบ หน้า 2 แทน

ภลสี (HCOLOR)

สั่งให้คอมพิวเตอร์เลือกสีที่ใช้แสดงแต่ละจุด ในภาวะการแสดงภาพละเอียด

วิธีเขียนคำสั่ง ภลสี n

n เป็นตัวเลข หรือนิพจน์ ก็ได้

n = 0 สีดำ	n = 4 สีดำ
n = 1 สีเขียว	n = 5 สีแสด
n = 2 สีม่วง	n = 6 สีฟ้า
n = 3 สี ขาว	n = 7 สีขาว

หลังจากระบุค่า n ในคำสั่ง **ภลสี** แล้ว คำสั่ง **ภลจุด** และคำสั่ง **วาด** จะให้ จุดและเส้นตามสีนั้นตลอด จนกว่าจะมีการระบุค่า n ใหม่

n ต้องมีค่าระหว่าง 0 ถึง 7 มิฉะนั้นเครื่องจะรายงานความผิดพลาดว่า "ปริมาณต้องห้าม"

ภาพที่เกิดขึ้นก่อนที่จะมีคำสั่ง **ภลสี** จะมีสีไม่แน่นอ คำสั่ง **ภลสี** ไม่มีผลต่อการแสดงภาพหยาบ ถ้าใช้คำสั่ง **ภลสี** เมื่อเครื่องยังไม่อยู่ในภาวะการแสดงภาพ จะไม่มีผลต่อการแสดงภาพละเอียดครั้งต่อไป

เมื่อผิด ไป (ONERR GOTO)

สั่งให้เครื่องไปทำ ณ เลขบรรทัดที่กำหนด หากพบข้อผิดพลาดหลังจากคำสั่งนี้

วิธีเขียนคำสั่ง เมื่อผิด ไป n

n เป็นเลขบรรทัด

ในระหว่างการทำงานตามโปรแกรม เมื่อมีความผิดพลาดเกิดขึ้นจะมีรหัสบอกชนิดของความผิดพลาด ไปเก็บไว้ที่หน่วยความจำตำแหน่ง 222 ซึ่งจะสามารถเรียกดูได้ด้วยคำสั่ง หยิบ (222) (ดูตารางข้อความบอกความผิดพลาด)

ตัวอย่างการใช้คำสั่งเมื่อผิดไปมีดังนี้

10 เมื่อผิด ไป 1000

(20-990) ตัวโปรแกรม

```
1000  ' โปรแกรมย่อยจัดการความผิดพลาด
1010  A = หยิบ (222)
1020  ถ้า A = 69 หรือ M = 163 แล้ว M = 0 : รับ M
1030  เดินต่อ
1040  กลับ
```

คำอธิบาย สมมติในโปรแกรมนี้มีคำสั่ง **รับ N$** และ **รับ M** อยู่ในระหว่างบรรทัด 20 ถึง 990

หากเกิดความผิดพลาดขึ้น เช่น

(ก) ป้อนอักษรแทนตัวเลขซึ่งเป็นข้อผิดพลาดแบบ "ข้อมูลผิดประเภท" (รหัส 163)

(ข) ป้อนค่าตัวเลขซึ่งใหญ่เกินไป หรือเล็กเกินไปซึ่งเป็นข้อผิดพลาดแบบ "ค่าตัวเลขเกินขอบเขต" (รหัส 69)

ทันทีที่เครื่องตรวจพบความผิดพลาด คำสั่งบรรทัด 10 จะส่งการทำงานไปที่โปรแกรมย่อย ที่เริ่มต้นที่ 1000

บรรทัด 1010 จะอ่านรหัสของความพลาด

1020 แก้ไขความผิดพลาดโดยให้รับค่า M ใหม่

1030 สั่งให้ทำโปรแกรมต่อ

1040 บอกให้ทราบว่าจบโปรแกรมย่อย ให้กลับโปรแกรมหลัก

มาตรา (SCALE)

เป็นคำสั่งที่ใช้กำหนดมาตราส่วนในการแสดงภาพละเอียด ที่เขียนขึ้นโดยใช้คำสั่ง **วาด** หรือ **วาดสี**

วิธีเขียนคำสั่ง มาตรา = n

n เป็นเลขจำนวนเต็ม หรือนิพจน์ที่มีค่าเป็นเลขจำนวนเต็ม

n เป็นตัวคูณสำหรับกำหนดขนาดของภาพที่แสดงบนจอ เช่น

มาตรา = 1 จะให้ภาพบนจอ มีขนาดเท่ากับที่ระบุไว้ในตารางรูปร่าง (SHAPE TABLE)

มาตรา = 2 จะให้ภาพบนจอที่มีขนาดใหญ่ขึ้นเป็น 2 เท่า

มาตรา = 0 จะมีผลเช่นเดียวกับ มาตรา = 255 คือให้ภาพขยาด 255 เท่า

ถ้า N มีค่าเกิน 255 เครื่องจะรายงานความผิดพลาดว่า "ปริมาณต้องห้าม"

ย้ายดิ่ง (VTAB)

เคลื่อนย้ายเคอร์เซอร์ตามแนวดิ่ง ไปยังบรรทัดที่ระบุ โดยที่ยังอยู่ในคอลัมน์เดิม

วิธีเขียนคำสั่ง ย้ายดิ่ง r

r = หมายเลขบรรทัดบนจอ (1 ถึง 24)

1 คือบรรทัดบนสุด และ 24 คือบรรทัดล่างสุด

ย้ายราบ (HTAB)

เคลื่อนย้ายเคอร์เซอร์ตามแนวราบ ไปยังคอลัมน์ที่ระบุ โดยที่ยังอยู่ในบรรทัดเดิม

วิธีเขียนคำสั่ง ย้ายราบ c

c = หมายเลขคอลัมน์ (1-40)

1 คือคอลัมน์ซ้ายสุด และ 40 คือคอลัมน์ขวาสุด

รอ (WAIT)

หยุดการทำงานของโปรแกรม จนกว่าหน่วยความจำที่ระบุ จะเป็นไปตามเงื่อนไขที่ระบุ

วิธีเขียนคำสั่ง รอ m, b, c

m เป็นตำแหน่งหน่วยความจำ

b เป็นเลขจำนวนเต็ม (0 ถึง 255) ซึ่งเมื่อแปลงเป็นเลขฐาน 2 จะเป็นตัวบอกว่าให้พิจารณาบิทใดของหน่วยความจำ ทั้งนี้บิทที่เป็น 1 จะถูกพิจารณา บิทที่เป็น 0 จะไม่ถูกพิจารณา

c เป็นเลขจำนวนเต็ม (0 ถึง 255) ซึ่งเมื่อแปลงเป็นเลขฐาน 2 แล้วจะเป็นเงื่อนที่กำหนดให้แต่ละบิทเป็น 1 หรือ 0

เมื่อพบคำสั่ง **รอ** เครื่องจะหยุดการทำงานของโปรแกรมและทำการตรวจสอบบิทที่ถูกระบุให้พิจารณาโดย b จนกว่าจะพบว่าบิทเหล่านั้นเป็นไปตามเงื่อนไขที่ระบุใน c แล้วจึงจะทำงานต่อ

ถ้า b และ c มีค่าเกิน 255 เครื่องจะรายงานความผิดพลาดว่า

"ปริมาณต้องห้าม"

ถ้า m เป็นตำแหน่งที่เกินหน่วยความจำที่มีอยู่ หรือเป็นตำแหน่งที่แทนอุปกรณ์ภายนอกที่ไม่พร้อม เครื่องจะชงักงัน (แฮ็ง) ซึ่งจะต้องทำการ รีเซ็ท

รับ (INPUT)

รับตัวอักษรจากแป้นพิมพ์ หรืออุปกรณ์ป้อนข้อมูลเข้าอย่างอื่น แล้วตรวจดูว่าเป็นข้อมูลที่ใช้การได้หรือไม่ ถ้าใช้การได้ ก็จะนำไปกำหนดเป็นค่าของตัวแปรที่มากับคำสั่ง **รับ** นั้น

วิธีเขียนคำสั่ง รับ ["ข้อความ";] x1[,x2,...,xn]

หรือ รับ ["ข้อความ";] x1$[,x2$,...xn$]

"ข้อความ" คือข้อความที่ต้องการให้แสดงบนจอ เพื่อให้ผู้ใช้เครื่องทราบว่าขณะนั้นเครื่องกำลังรอรับข้อมูลตัวใด

x1, x2, ...,xn เป็นชื่อตัวแปรเลขใด ๆ

x1$, x2$, ...,xn$ เป็นชื่อตัวแปรอักขระใด

จะใช้ x และ x$ ปนกันในคำสั่งเดียวก็ได้

ถ้าไม่มี "ข้อความ" เครื่องจะพิมพ์เครื่องหมายคำถามบนจอ

ถ้ามี "ข้อความ" เครื่องจะพิมพ์ข้อความบนจอสำหรับตัวแปรตัวแรกครั้ง**เดียว** จะไม่มีการพิมพ์สำหรับตัวแปรตัวอื่น ๆ และจะไม่มีการพิมพ์เครื่องหมายคำถาม

ในการป้อนข้อมูลสำหรับกรณีที่คำสั่ง **รับ** คำสั่งเดียว ต้องการข้อมูลหลายตัว เราจะป้อนแต่ละตัวแล้วกด **รีเทอน** เพื่อให้ขึ้นบรรทัดใหม่ก็ได้ ในกรณีนี้จะมีเครื่องหมายคำถามคู่ (??) สำหรับข้อมูลแต่ละตัวที่ต้องการ หรือเราอาจป้อนข้อมูลหลายตัวในบรรทัดเดียวกันขั้นด้วย , ก็ได้

ถ้าข้อมูลใช้การไม่ได้ เป็นต้นว่าป้อนตัวอักษรในกรณีที่ต้องการตัวเลข เครื่องจะบอกว่า "ป้อนใหม่" ซึ่งเราจะต้องป้อนข้อมูลตัวนั้นหรือชุดนั้นใหม่ทั้งหมด ทั้งนี้เพราะ เมื่อมีคำว่า "ป้อนใหม่" ปรากฏขึ้นบนจอ เครื่องจะย้อนกลับไปทำคำสั่ง รับใหม่ตั้งแต่ต้น โดยแสดงเครื่องหมายคำถาม หรือ "ข้อความ" อีกครั้งหนึ่ง

คำสั่ง **รับ** ที่ต้องการตัวเลข ถ้ากด **รีเทอน** เปล่า ๆ โดยไม่มีการป้อนตัวเลข เครื่องก็จะบอกว่า "ป้อนใหม่" ตัวเลขที่ ไทยปกาสิตยอมรับคือ เลขจำนวนเต็ม หรือเลขจำนวนจริง (มีจุดทศนิยมได้) อาจมีเครื่องหมาย + หรือ -

นำหน้าหรือไม่ก็ได้ หรือตัวเลขที่เขียนแบบวิทยาศาสตร์โดยใช้ E

คำสั่ง **รับ** ที่ต้องการขบวนอักขระ ถ้ากด รีเทอน เปล่าๆ เครื่องจะถือว่าเป็นขบวนอักขระว่าง ("") ถ้ามีการใช้เครื่องหมาย ; อักขระทุกตัวที่ตามหลังเครื่องหมายนี้จะไม่ถูกรับไป เครื่องจะรับอักขระทุกตัวที่ตามหลังอัญญประกาศตัวแรกไปจนถึงอัญญประกาศตัวถัดไป หรือรีเทอน

คำสั่ง **รับ** ห้ามใช้เป็นคำสั่งตรง

เรียก (CALL)

สั่งให้ไปทำงานตาม **โปรแกรมภาษาเครื่อง** (machine language subroutine) ซึ่งถูกเก็บไว้ ณ ตำแหน่งที่กำหนดของหน่วยความจำ

วิธีเขียนคำสั่ง เรียก m

m = ตำแหน่งหน่วยความจำ

เราอาจใช้กับโปรแกรมภาษาเครื่องที่เขียนขึ้นเอง หรือกับโปรแกรมภายในที่ใช้ควบคุมการทำงานของเครื่อง (MONITOR SUBROUTINES) ก็ได้

เรียกคืน (RECALL)

สั่งให้อ่านค่าตัวแปรแบบเลขจากเทปตลับ

วิธีเขียนคำสั่ง เรียกคืน v

v = ชื่อตัวแปรแถวแบบเลขซึ่งจะต้องมีรูปแบบดังนี้

v (i, j,...) โดยที่ i, j ฯลฯ เป็นเลขลำดับในแต่ละมิติ

คำสั่งนี้สั่งให้รอรับค่าตัวเลขจากเทปตลับ ซึ่งจะเปิดเครื่องเล่นเทปในภาวะเล่นรออยู่ก่อนแล้ว เครื่องจะส่งเสียงดังปิ๊บ เมื่อเริ่มรับข้อมูล และเมื่อจบการรับข้อมูล ตัวแปรแถวที่ถูกอ้างถึงในคำสั่งนี้จะต้องมีการกำหนดขนาดไว้ล่วงหน้า (ดูคำสั่ง **ขนาด**) มิฉะนั้นเครื่องจะรายงานว่า "หมดข้อมูลแล้ว"

การเรียกชื่อตัวแปร อาจเรียกแตกต่างไปจากชื่อที่ใช้ในคำสั่ง **เก็บ** (STORE) ก็ได้ แต่ **ขนาด** ที่ระบุก่อนจะ **เรียกคืน** ต้องเท่ากับขนาดที่ระบุก่อนการ **เก็บ** มิฉะนั้นจะเกิดความผิดพลาดขึ้น

คำสั่งนี้ไม่สามารถใช้อ่านค่าตัวแปรแถวแบบอักขระได้ แต่ค่าตัวเลขที่อ่านมาได้อาจแปลงเป็นตัวอักษรได้ โดยใช้คำสั่ง **รหัส**

ล้าง (CLEAR)

สั่งให้ล้างค่าเดิมของตัวแปร ทุกชนิด ทุกตัว

วิธีเขียนคำสั่ง ล้าง

ตัวแปรเลข และสมาชิกของตัวแปรแถวแบบเลข ทุกตัวจะถูกกำหนดค่าเป็น 0 ตัวแปรอักขระ และสมาชิกของตัวแปรแถวแบบอักขระ ทุกตัว จะถูกกำหนดค่าเป็นขบวนอักขระว่าง ("")

คำสั่งนี้ มีผลเสมือนหนึ่ง ปิดเครื่องแล้วเปิดใหม่ หลังจากคำสั่ง **ล้าง** โปรแกรมจะทำงานต่อไปได้ ถ้าการล้างค่าตัวแปรต่าง ๆ ไม่ทำให้ขั้นตอนของการดำเนินโปรแกรมผิดไป

ล้างจอ (HOME)

ทำการล้างจอแล้ววางเคอร์เซอร์ที่มุมบนซ้าย (บรรทัด 1 คอลัมน์ 1)

วิธีเขียนคำสั่ง ล้างจอ

เลิกตาม (NOTRACE)

บอกเลิกใช้คำสั่ง **ตาม**

วิธีเขียนคำสั่ง เลิกตาม

วาด (DRAW)

สั่งให้เขียนรูปร่างในลักษณะของภาพละเอียด

วิธีเขียนคำสั่ง วาด n [ที่ x, y]

n = หมายเลขประจำตัวของรูปร่างที่เก็บไว้ในตารางรูปร่าง (n = 1 ถึง 255) อาจใช้นิพจน์แทน n ได้

x = ตำแหน่งบนจอตามแนวราบ (x = 0 ถึง 191)

y = ตำแหน่งบนจอตามแนวดิ่ง (y = 0 ถึง 279)

สีของภาพกำหนดโดยคำสั่ง **ภลสี** ล่าสุด การกำหนด **ขนาด** และมุมของการ **หมุน** ภาพ ต้องใช้คำสั่ง **ขนาด** และ **หมุน** ก่อนสั่ง **วาด**

ตำแหน่ง x, y เป็นตำแหน่งเริ่มต้นของการ **วาด** รูปร่างที่กำหนด ถ้าไม่ระบุ x, y การวาดจะเริ่มต้นที่ตำแหน่งสุดท้ายของการทำตามคำสั่ง วาด

วาดสี หรือ ภลจุด

อย่าใช้คำสั่งนี้โดยไม่มีตารางรูปร่างเก็บอยู่ในหน่วยความจำ เพราะอาจทำให้เครื่องชงักงันได้ หรืออาจได้ภาพที่ไม่มีความหมาย ถ้าโปรแกรมมีความยาวมาก จนทับหน่วยความจำส่วนของภาพละเอียดแล้ว การวาดบางส่วนหรือทั้งหมดอาจเสียได้

วาดสี (XDRAW)

เขียน หรือ ลบ รูปร่างในลักษณะของภาพละเอียด โดยใช้คู่สีตรงข้าม

วิธีเขียนคำสั่ง วาดสี n [ที่ x, y]

n = หมายเลขประจำตัวของรูปร่างที่เก็บไว้ในตารางรูปร่าง (n = 1 ถึง 255) อาจใช้นิพจน์แทน n ได้

x = ตำแหน่งบนจอตามแนวราบ (x = 0 ถึง 191)

y = ตำแหน่งบนจอตามแนวดิ่ง (Y = 0 ถึง 279)

เมื่อใช้คำสั่งนี้ เครื่องจะเขียนรูปร่างตามหมายเลขที่ระบุซึ่งเก็บไว้ในตารางรูปร่าง ในการเขียนนี้ จะกำหนดสีของแต่ละจุดให้เป็นสีคู่ตรงข้ามกับสีที่ปรากฏ ณ จุดนั้นในขณะนั้น การจัดคู่สีตรงข้ามเป็นดังนี้

0	(ดำ)	ตรงข้ามกับ	3	(ขาว)
1	(เขียว)	"	2	(ม่วง)
4	(ดำ)	"	7	(ขาว)
5	(แสด)	"	6	(ฟ้า)

ขนาดและมุมการหมุนของภาพจะต้องกำหนดล่วงหน้าด้วยคำสั่ง **มาตรา** และ **หมุน**

คำสั่ง **วาดสี** มีข้อได้เปรียบกว่าคำสั่ง **วาด** คือสามารถลบภาพได้ง่าย เพราะเพียงแต่ใช้คำสั่งวาดสีที่มีค่า n, x, y ซ้ำอีกครั้ง เครื่องจะเขียนสีคู่ตรงข้าม ทับลงไปจุดต่อจุด ซึ่งก็คือ การลบภาพนั้นเอง ในขณะส่วนอื่นๆ ยังคงเดิม

ถ้าไม่ระบุ x, y การวาดจะเริ่มจากตำแหน่งสุดท้ายของการทำตามคำสั่ง **วาด, วาดสี, ภลจุด** แต่ถ้าระบุค่า x, y การวาดจะเริ่มจากจุดนั้

สลับแสง (INVERSE)

เปลี่ยนสีพื้นของจอและสีของตัวอักษร จากพื้นดำตัวขาวเป็นพื้นขาวตัวดำ

วิธีเขียนคำสั่ง สลับแสง

สลับแสง มีผลต่อการแสดงอักษร **ภาษาอังกฤษเท่านั้น** ไม่สามารถใช้กับอักษรภาษาไทยได้ หลังจากที่เครื่องพบคำสั่ง **สลับแสง** แล้ว การแสดงบนจอที่มาจากคำสั่ง **พิมพ์** ที่ตามขึ้นภายหลัง จะเป็นแบบพื้นขาวตัวดำ แต่ตัวอักษรที่มาจากการป้อนข้อมูลให้แก่คำสั่ง รับ จะเป็นแบบธรรมดา อย่าใช้คำสั่งนี้ในระหว่างการส่งข้อมูลไปยังเครื่องบันทึกข้อมูล หรือเครื่องพิมพ์ เพราะจะทำให้รหัสที่ส่งไปไม่ถูกต้อง

ใช้คำสั่ง **ปกติ** เพื่อกลับคืนสู่การแสดงภาพแบบธรรมดา

สี (COLOR)

กำหนดสีของภาพหยาบ

วิธีเขียนคำสั่ง สี = n

(n = 0 ถึง 15)

ภายหลังจากที่กำหนดสีด้วยคำสั่งนี้ไปแล้ว คำสั่ง **จุด เส้นนอน เส้นดิ่ง** จะใช้สีตามที่กำหนดตลอดไป จนกว่าจะมีการกำหนดสีใหม่ ถ้าไม่มีการกำหนด เครื่องจะถือว่าเป็นสีดำ (N=0)

คำสั่งนี้ ไม่มีผลต่อการแสดงภาพละเอียด หากใช้คำนี้ในภาวะจอปกติ (สำหรับแสดงข้อความ) จะทำให้คำสั่ง **จุด** แสดงตัวอักษรแทนจุด

เส้นดิ่ง (VLIN)

ขีดเส้นตามแนวดิ่งบนจอในลักษณะภาพหยาบ

วิธีเขียนคำสั่ง เส้นดิ่ง y1, y2 ที่ x

(x, y1) = ตำแหน่งจุดเริ่มต้นของเส้น

(x, y2) = ตำแหน่งจุดสิ้นสุดของเส้น

x = 0 ถึง 39

y = 0 ถึง 47

สีของเส้นจะถูกกำหนดโดยคำสั่ง **สี** ล่าสุด ถ้าใช้คำสั่ง **เส้นดิ่ง**ในภาวะจอปกติที่แสดงข้อความ หรือในกรณีที่เส้นล้ำเข้าไปในบริเวณ 4 บรรทัดล่างที่ไปแสดงข้อความ บางส่วนหรือทั้งหมดของเส้นจะปรากฏเป็นตัวอักษรแทนจุดสี

เส้นนอน (HLIN)

ขีดเส้นตามแนวนอนบนจอในลักษณะภาพหยาบ

วิธีเขียนคำสั่ง เส้นนอน x1, x2 ที่ y

(x1, y) = ตำแหน่งจุดเริ่มต้นของเส้น

(x2, y) = ตำแหน่งจุดสิ้นสุดของเส้น

x = 0 ถึง 39

y = 0 ถึง 47

รายละเอียดอื่นๆ เช่นเดียวกับ **เส้นดิ่ง**

หมายเหตุ (REM)

ใช้เขียนข้อความที่ไม่ใช่คำสั่งลงในโปรแกรม เพื่อประโยชน์ในการเตือนความจำ หรือบอกหน้าที่ หรือภาวะของส่วนต่าง ๆ ของโปรแกรม

วิธีเขียนคำสั่ง ใช้เครื่องหมาย ' แทนคำว่าหมายเหตุ

(ก) เขียนเป็นบรรทัดต่างหาก เช่น

120 ' โปรแกรมคำนวณภาษีเงินได้

(ข) เขียนต่อท้ายคำสั่งอื่นในบรรทัดเดียวกัน เช่น

200 จาก N = 1 ถึง 12: 'คำนวณรายได้แต่ละเดือน

ข้อควรจำ เครื่องจะถือว่าทุกอย่างที่ตามหลังเครื่องหมาย ' เป็นหมายเหตุ ฉะนั้นอย่าเขียนคำสั่งใด ๆ ต่อท้าย

หมุน (ROT)

สั่งให้หมุนภาพที่วาดไว้ด้วยคำสั่ง **วาด** หรือ **วาดสี** ในสภาวะภาพละเอียด

วิธีเขียนคำสั่ง หมุน n

n เป็นเลขจำนวนเต็ม หรือนิพจน์ที่มีค่าเป็นเลขจำนวนเต็ม เฉพาะบางค่าที่มีความหมายเท่านั้น ขึ้นกับขนาดของภาพที่กำหนด โดยคำสั่ง **มาตรา** ด้วยดังนี้

มาตรา = 1	n = 0	มุม 0 องศา
	n = 16	มุม 90 องศา
	n = 32	มุม 180 องศา
	n = 48	มุม 270 องศา

(4 ค่า)

มาตรา = 2	n = 0	มุม 0 องศา
	n = 8	มุม 45 องศา
	n = 16	มุม 90 องศา
	n = 24	มุม 135 องศา
	n = 32	มุม 180 องศา
	n = 40	มุม 225 องศา
	n = 48	มุม 270 องศา
	n = 56	มุม 315 องศา

(8 ค่า)

ฯลฯ

ความละเอียดสูงสุดที่เป็นไปได้ คือ เมื่อมาตรา = 5 จะมีค่า n ถึง 64 ค่า (n = 0 ถึง n = 63) แต่ละค่าจะหมุนเป็นมุมเพิ่มขึ้น 5.625 องศา

ถ้ามีการระบุค่า n ที่ไม่มีความหมาย เครื่องจะยึดถือค่าต่ำลงไปค่าแรกที่มีความหมาย ค่า n = 64 มีความหมายเช่นเดียวกับ n = 0 และค่าที่เกิน 64 หมายถึงการหมุนเกิน 1 รอบ แต่ถ้าระบุ n เกิน 255 เครื่องจะรายงานความผิดพลาดว่า "ปริมาณต้องห้าม"

หยอด (POKE)

สั่งให้บรรจุข้อมูลขนาด 1 ไบท์ ลงในตำแหน่งที่ระบุในหน่วยความจำ

วิธีเขียนคำสั่ง หยอด a, n

a = เลขจำนวนเต็มที่แทนตำแหน่งของหน่วยความจำ

n = เลขจำนวนเต็ม (0 ถึง 255) ซึ่งเป็นค่าตัวเลขของข้อมูล 8 บิท เมื่อแปลงเป็นเลขฐาน 10

ถ้า a มีค่าเกินขนาดของหน่วยความจำ หรือเป็นตำแหน่งที่อ้างถึงอุปกรณ์ภายนอกที่ไม่พร้อม คำสั่งนี้ไม่มีผล ข้อควรระวังในการใช้คำสั่งนี้ คือ อย่าบรรจุข้อมูลลงไปในบริเวณหน่วยความจำที่ใช้ควบคุมการทำงานของคอมพิวเตอร์ หรือใช้บรรจุระบบแปลภาษา หรือตัวโปรแกรม มิฉะนั้นอาจทำให้เกิดอาการชงักงัน หรืออาจมีการทำลายโปรแกรมได้

หยิบ **(PEEK)**

เป็นคำสั่งประเภทฟังก์ชั่น ดูข้อ 5.2

หยุด **(STOP)**

สั่งให้คอมพิวเตอร์หยุดทำงานตามโปรแกรมที่กำลังทำอยู่ทันที

วิธีเขียนคำสั่ง หยุด

คอมพิวเตอร์จะแสดงข้อความว่า "ชงักที่ n " โดยที่ n เป็นเลขบรรทัดของคำสั่งสุดท้ายก่อนหยุด จากนั้นคอมพิวเตอร์จะรอรับคำสั่ง

ให้ **(LET)**

สั่งให้กำหนดค่าตัวแปร

วิธีเขียนคำสั่ง

(ก) **ให้** x = e หรือ **ให้** x$ = e$

(ข) x = e หรือ x$ = e$

(ไม่ต้องใส่คำว่า "ให้")

x เป็นชื่อตัวแปร

c เป็นตัวเลข หรือนิพจน์

x$ เป็นชื่อตัวแปรอักขระ

e$ เป็นนิพจน์ของตัวแปรอักขระ

ใหม่ **(NEW)**

สั่งให้ลบโปรแกรม และตัวแปรทุกตัวจากหน่วยความจำ

วิธีเขียนคำสั่ง ใหม่

คำสั่งนี้จะกำหนดค่า **ขอบจำล่าง** ใหม่ไว้ที่ระดับต่ำสุดของหน่วยความจำที่ยังไม่ได้ใช้ แต่จะไม่มีผลต่อ **ขอบจำบน** , **สี** และ **ภลสี**

อ่าน **(READ)**

สั่งให้อ่านข้อมูลจากคำสั่ง "ข้อมูล"

วิธีเขียนคำสั่ง อ่าน x1 [x2, x3,...]

หรือ อ่าน x1$, [x2$, x3$,...]

คำสั่ง **อ่าน** จะใช้คู่กับข้อมูลเสมอ โดยคำสั่งทั้งสองนี้จะจับคู่ชื่อตัวแปรกับข้อมูลเป็นคู่ ๆ ไป และกำหนดค่าของตัวแปรเท่ากับข้อมูลนั้น คู่ที่ใช้แล้วจะไม่มีการใช้ซ้ำ เว้นแต่จะถูกสั่งให้เริ่มต้นใหม่ด้วยคำสั่ง **สงวน**

ชื่อตัวแปรกับข้อมูลต้องเป็นชนิดเดียวกัน ถ้าผิดชนิด เครื่องจะรายงานความผิดพลาดว่า"ไวยากรณ์ผิด" (แม้ว่าตัวแปรอักขระสามารถอ่านข้อมูลที่เป็นตัวเลขได้ แต่จะถือว่าตัวเลขชุดนั้นเป็นขบวนอักขระ)

ถ้าคำสั่ง **อ่าน** มีจำนวนชื่อตัวแปรมากกว่าจำนวนข้อมูล เครื่องจะรายงานความผิดพลาดว่า "หมดข้อมูลแล้ว"

สำหรับคำสั่ง **อ่าน** ที่เป็นคำสั่ง DOS ให้ดูคู่มือ DOS

5.3 ฟังก์ชั่น

ฟังก์ชั่นคือ สูตรสำเร็จที่มีอยู่ภายในคอมพิวเตอร์ ซึ่งจะเขียนในรูป f(x) โดยที่ f เป็นชื่อของฟังก์ชั่น และ x เป็นตัวแปร ฟังก์ชั่นบางฟังก์ชั่นอาจมีตัวแปรได้มากกว่า 1 ตัว การเรียกใช้ฟังก์ชั่นทำได้โดยการเขียนคำสั่ง ให้ชื่อฟังก์ชั่นเป็นส่วนหนึ่งของคำสั่ง **พิมพ์** หรือคำสั่งกำหนดค่า พร้อมกับระบุค่าของตัวแปรเป็นตัวเลขโดยตรง หรือระบุค่าตัวแปรเป็นนิพจน์ก็ได้ เช่น

```
10   พิมพ์  SIN  (30)
20   N = SQR (M)
```
เป็นต้น

ในข้อนี้จะกล่าวถึงรายละเอียดของฟังก์ชั่นแต่ละตัว เรียงตามลำดับอักษร (ภาษาไทยก่อนภาษาอังกฤษ)

กลาง (MID$)

แยกส่วนกลางของขบวนอักขระ

วิธีเขียนคำสั่ง กลาง (a$, m, n)

m, n เป็นตัวเลขจำนวนเต็ม (เป็นนิพจน์ได้)

แยกส่วนกลางของตัวแปรอักขระชื่อ a$ ตั้งแต่ตัวอักษรตัวที่ m เป็นต้นไปจำนวน n ตัว

ขวา (RIGHT$)

แยกส่วนขวาของขบวนอักขระ

วิธีเขียนคำสั่ง ขวา (a$, n)

n เป็นเลขจำนวนเต็ม (1 ถึง 255) เป็นนิพจน์ได้

แยกส่วนขวาของตัวแปรอักขระชื่อ a$ จำนวน n ตัว (นับจากตัวขวาสุด)

ความยาว (LEN)

หาค่าความยาวของขบวนอักขระ โดยบอกเป็นจำนวนตัวอักษร ที่อยู่ในขบวนอักขระนั้น

วิธีเขียนคำสั่ง ความยาว (a$)

a$ = ชื่อตัวแปรอักขระ

การนับจำนวนตัวอักษร จะนับรวมทั้งเว้นวรรค และตัวอักษรที่ไม่ปรากฏบนจอด้วย ค่าที่หาได้ต้องไม่เกิน 255 มิฉะนั้นเครื่องจะรายงานความผิดพลาดว่า "ขบวนอักขระยาวเกิน"

คันบังคับ (PDL)

หาค่าตัวเลขจากคันบังคับ

วิธีเขียนคำสั่ง คันบังคับ (n)

n = 0 ถึง 3 เป็นเลขประจำตัวของคันบังคับซึ่งสามารถนำมาต่อกับคอมพิวเตอร์ได้ 4 อัน

ตัวเลขที่หาได้มีค่าระหว่าง 0 ถึง 255 ขึ้นอยู่กับค่าของความต้านทานที่ปรากฏระหว่างขั้วต่อสายสำหรับคันบังคับหมายเลขนั้น ๆ

ค่า (VAL)

หาค่าของขบวนอักขระซึ่งประกอบด้วยตัวเลข

วิธีเขียนคำสั่ง ค่า (n$)

n$ = ขบวนอักขระที่ประกอบด้วยตัวเลข เขียนในรูปแบบของเลขจำนวนเต็ม เลขจำนวนจริง หรือเลขแบบวิทยาศาสตร์

ค่าที่หาได้ต้องไม่เกินขอบเขตของเลขจำนวนจริง มิฉะนั้นเครื่องจะบอกสัญญาณความผิดพลาดว่า "ค่าเกินขอบเขต"

เครื่องหมาย (SGN)

บอกเครื่องหมายของจำนวนเลข

วิธีเขียนคำสั่ง เครื่องหมาย (n)

n = จำนวนเลข หรือนิพจน์

ถ้าค่าที่หาได้เป็น + 1 แสดงว่า n เป็นบวก

- 1 แสดงว่า n เป็นลบ

0 แสดงว่า n เป็นศูนย์

ซ้าย (LEFT$)

แยกส่วนซ้ายของขบวนอักขระ

วิธีเขียนคำสั่ง ซ้าย (a$, n)

n เป็นเลขจำนวนเต็ม (1 ถึง 255) เป็นนิพจน์ได้

แยกส่วนซ้ายของตัวแปรชื่อ a$ จำนวน n ตัว (นับจากตัวซ้ายสุด)

ตำแหน่ง (POS)

อ่านค่าตำแหน่งของเคอร์เซอร์

วิธีเขียนคำสั่ง ตำแหน่ง (n)

n เป็นตัวแปรเทียมจะมีค่าใดก็ได้ระหว่าง 0 กับ 255

คำสั่งนี้ จะอ่านค่าตำแหน่งคอลัมน์ของเคอร์เซอร์ (0 ถึง 39) ณ ขณะนั้น

แปลง (STR$)

แปลงค่าตัวเลขเป็นขบวนอักขระ

วิธีเขียนคำสั่ง แปลง (n)

n เป็นค่าตัวเลข หรือนิพจน์ใดๆ

ผลจากคำสั่งนี้ คือจะได้ขบวนอักขระเหมือนกับที่จะปรากฏบนจอ เมื่อสั่งด้วยคำสั่งพิมพ์ เช่น

แปลง (2/3) จะให้ขบวนอักขระเป็น .666666667

แปลง (2468013579) จะให้ขบวนอักขระเป็น 2.46801358E +09

ถ้า n มีค่าเกินขอบเขตของเลขจำนวนจริง เครื่องจะรายงานความผิดพลาดว่า "ค่าเกินขอบเขต"

รหัส (ASC)

หาค่ารหัส ASC11 ของตัวอักษร

วิธีเขียนคำสั่ง รหัส (e$)

e$ เป็นตัวอักษรเดี่ยว หรือนิพจน์ซึ่งมีค่าเป็นตัวอักษรเดี่ยว (ดูตารางรหัส ASC11 ในภาคผนวก)

เรียกใช้ (USR)

แยกจากโปรแกรมหลักไปทำโปรแกรมย่อยภาษาเครื่อง โดยนำค่าตัวเลข ใช้ในการคำนวณด้วย

วิธีเขียนคำสั่ง เรียกใช้ (n)

n = ค่าตัวเลขหรือนิพจน์ซึ่งจะให้ผลลัพธ์เป็นค่าตัวเลข

เมื่อพบคำสั่งนี้ เครื่องจะนำตัวเลข n ไปเก็บที่หน่วยความจำ $9D ถึง $A3 (หน่วยความจำส่วนนี้ใช้สำหรับเก็บตัวเลขจำนวนจริงที่ใช้ในการคำนวณ มีชื่อเรียกว่า (floating-point accumuletor) จากนั้นเครื่องจะแยกไปสู่โปรแกรมย่อยที่ตำแหน่ง $0A ตำแหน่ง $0A ถึง $0C เป็นที่เก็บคำสั่งภาษาเครื่องจำนวน 3 ไบท์ ซึ่งจะสั่งให้เครื่องกระโดด (JMP) ไปทำโปรแกรมย่อยภาษาเครื่องตามตำแหน่งที่ระบุในไบท์ที่ 2 และ 3 โปรแกรมย่อยภาษาเครื่องนี้สามารถนำค่า n ไปใช้ในการคำนวณ แล้วนำผลลัพธ์ของการคำนวณไปเก็บที่ $9D ถึง $A3 เมื่อจบโปรแกรมย่อยคำสั่ง RTS จะส่งการทำงานกลับมาที่โปรแกรมหลัก

เลื่อน (TAB)

เลื่อนเคอร์เซอร์ไปทางขวาไปยังตำแหน่งที่ระบุ

วิธีเขียนคำสั่ง เลื่อน (n)

n = ตัวเลข 1 ถึง 255 อาจเป็นนิพจน์ก็ได้

ตามปกติใช้ n = 1 ถึง 40 สำหรับจอ 40 คอลัมน์ ถ้าค่า n เกินค่าจำนวนคอลัมน์สูงสุด การเลื่อนจะข้ามบรรทัด และนับจำนวนช่องตัวอักษรต่อกัน ข้อควรทราบเกี่ยวกับคำสั่งนี้ก็คือ

1. คำสั่งนี้ไม่สามารถสั่งให้เคอร์เซอร์เลื่อนถอยหลัง หรือเลื่อนไปทางซ้ายได้
2. การเลื่อนไปทางขวา เป็นการพิมพ์ช่องว่าง ซึ่งจะลบตัวอักษรที่ถูกเลื่อนผ่าน

ว่าง (FRE)

ให้หาจำนวนไบท์ของหน่วยความจำที่ยังว่างอยู่ขณะนั้น ในระหว่างการใช้โปรแกรม

วิธีเขียนคำสั่ง ว่าง (n)

n เป็นตัวแปรเทียมซึ่งจะมีค่าใดก็ได้ (0 ถึง 255)

หน่วยความจำส่วนที่ว่าง คือส่วนที่อยู่ใต้ส่วนที่ใช้เก็บขบวนอักขระ และอยู่เหนือส่วนที่ใช้เก็บตัวแปรแถว ถ้าจำนวนไบท์มีมากกว่า 32767 ค่าที่หาได้จะเป็นเลขติดลบ ให้นำเลขติดลบนั้นมาบวกด้วย 65536 จะได้ค่าจำนวนไบท์ว่างที่แท้จริง

คำสั่ง **ว่าง** ยังมีหน้าที่อีกอย่างคือ ทำการลบขบวนอักขระที่เลิกใช้แล้วทิ้งไป ตามปกติถ้ามีการเปลี่ยนค่าของขบวนอักขระในโปรแกรม ค่าเก่ายังเก็บอยู่ในหน่วยความจำ และค่าใหม่ก็ถูกเพิ่มเข้าไป ในที่สุด ส่วนที่เก็บขบวนอักขระนี้ อาจล้นออกไปทับส่วนอื่น ทำให้การทำงานของโปรแกรมเสียไป เราแก้ไขได้โดยการสั่ง A = ว่าง (0) เป็นครั้งคราว ในกรณีที่โปรแกรมมีการใช้ขบวนอักขระมาก

เว้นวรรค (SPC)

เลื่อนเคอร์เซอร์ไปทางขวาตามจำนวนช่องตัวอักษร

วิธีเขียนคำสั่ง พิมพ์ เว้นวรรค (n)

n = เลขจำนวนเต็ม (1 ถึง 255) หรือนิพจน์

คำสั่งนี้ต้องใช้คู่กับคำสั่ง **พิมพ์** การเลื่อนจะเป็นการเลื่อนไปทางขวาเท่านั้น และลบตัวอักษรทุกตัวที่เลื่อนผ่าน ให้สังเกตว่า คำสั่งนี้กำหนดจำนวนช่องที่จะให้เลื่อนไป ต่างจากคำสั่ง **เลื่อน** ซึ่งกำหนดตำแหน่งปลายทางของการเลื่อน

สัมบูรณ์ (ABS)

หาค่าสัมบูรณ์ (คือค่าตัวเลขโดยไม่ติดเครื่องหมาย)

วิธีเขียนคำสั่ง สัมบูรณ์ (n)

สุ่ม (RND)

สุ่มตัวเลขขึ้นมา 1 ตัว

วิธีเขียนคำสั่ง สุ่ม (n)

n = ค่าตัวเลข (0 ถึง 255) หรือนิพจน์

ตัวเลขที่สุ่มมาได้มีค่าระหว่าง 0 ถึง 1 เสมอ ถ้าใช้คำสั่ง **สุ่ม**(0) จะได้ตัวเลขตัวใหม่ที่สุ่มได้ล่าสุด

ถ้า n เป็นลบเป็นการระบุว่าให้ใช้เลขสุ่มชุดเดิม โดยค่า n หนึ่ง ๆ จะให้เลขตัวเดียวกันทุกครั้ง และเมื่อสั่งด้วย n ที่เป็นลบไปครั้งหนึ่งแล้ว แม้ต่อมาจะสั่งด้วย n ที่เป็นบวก ก็จะเป็นการนำเลขชุดเดิมมาใช้ จนกว่าจะสั่ง **สุ่ม** (0)

ถ้า n เป็นบวก (โดยยังไม่เคยสั่งด้วย n ที่เป็นลบมาก่อน) จะเป็นการสุ่มเลขตัวใหม่ทุกครั้ง

สูตร (FN)

ดูกำหนดสูตร (DF FN) ในข้อ 5.1

หยิบ (PEEK)

อ่านค่าที่เก็บในหน่วยความจำ ณ ตำแหน่งที่ระบุ

วิธีเขียนคำสั่ง n = -32767 ถึง 32767

เลขบวกเป็นค่าตำแหน่งที่แท้จริงของหน่วยความจำระหว่าง 0 ถึง 32767 ส่วนเลขลบเป็นค่าที่ใช้แทนหน่วยความจำตั้งแต่ 36768 ถึง 65536 ดังนั้น -16384 หมายถึงหน่วยความจำ ณ ตำแหน่ง 49152 (มาจาก 65536 ลบด้วย 16384)

อักขร (CHR$)

ให้ตัวอักษรที่ตรงกับรหัส ASC ที่กำหนด

วิธีเขียนคำสั่ง n = 1 ถึง 255 อาจเป็นนิพจน์ก็ได้

ถ้า n เกิน 255 เครื่องจะรายงานความผิดพลาดว่า "ปริมาณต้องห้าม"

ATN

คำนวณค่า arctan ของตัวเลข

วิธีเขียนคำสั่ง ATN (n)

n = เลขจำนวนจริงอาจเป็นนิพจน์ได้

ค่ามุมที่คำนวณได้อยู่ในหน่วยเรเดียน โดยมีค่าในช่วงระหว่าง - PI/2 ถึง PI/2

COS

คำนวณค่า COS ของมุม

วิธีเขียนคำสั่ง COS (n)

n = เลขจำนวนจริง ซึ่งแสดงค่ามุมในหน่วยเรเดียน อาจเป็นนิพจน์ก็ได้

EXP

คำนวณค่า e ยกกำลัง

วิธีเขียนคำสั่ง EXP (n)

n = เลขจำนวนจริง อาจเป็นนิพจน์ได้

คำสั่งนี้ คำนวณหาค่า e ยกกำลัง n โดยที่ e คือฐานของ natural log ซึ่งมีค่า 2.71828183

INT

ให้เลขจำนวนเต็มจากเลขที่กำหนด (โดยการตัดเศษที่อยู่หลังจุดทศนิยมทิ้ง)

วิธีเขียนคำสั่ง INT (n)

n = เลขจำนวนจริงอาจเป็นนิพจน์ได้

LOG

คำนวนค่า natural log (log ฐาน e) ของเลขที่กำหนด

วิธีเขียนคำสั่ง LOG (n)

n = เลขจำนวนจริง อาจเป็นนิพจน์ได้

SIN

คำนวณค่า sin ของมุม

วิธีเขียนคำสั่ง SIN (n)

n = เลขจำนวนจริงซึ่งแสดงค่ามุมในหน่วยเรเดียน อาจเป็นนิพจน์ก็ได้

SQR

หาค่ารากที่สองของเลขบวก

วิธีเขียนคำสั่ง SQR (n)

n = เลขจำนวนจริงที่เป็นบวก อาจเป็นนิพจน์ได้

TAN

หาค่า tan ของมุม

วิธีเขียนคำสั่ง TAN (n)

n = เลขจำนวนจริงแสดงค่ามุมในหน่วยเรเดียน อาจเป็นนิพจน์ก็ได้

ภาคผนวก

ผ 1

คำสงวนของไทยปกาสิต

กระพริบ	เดินต่อ	มาตรา	สีจอ	*
กลับ	โดด	เมื่อผิด	สุ่ม	/
กลาง	ตัด	ไม่	สูตร	^
กำหนด	ตาม	ย้ายดิ่ง	เส้นดิ่ง	>
เก็บ	ตำแหน่ง	ย้ายราบ	เส้นนอน	<
ขนาด	ถ้า	รหัส	หมุน	=
ขวา	ถึง	รอ	หยอด	'
ข้อมูล	ทำ	รับ	หยิบ	
ขอบจำบน	ทำต่อ	เรียก	หยุด	
ขอบจำล่าง	ทีละ	เรียกคืน	หรือ	
ขึ้นกับ	ที่	เรียกใช้	ให้	
เข้า	บรรจุ	ล้าง	ใหม่	
ความยาว	บรรจุภาพ	ล้างจอ	อักขร	
เครื่องหมาย	บันทึก	เลิกตาม	อ่าน	
ความเร็ว	ปกติ	เลื่อน	ATN	
คันบังคับ	แปลง	แล้ว	COS	
ค่า	ไป	และ	EXP	
จบ	ไปภาค	วาด	INT	
จับ	พิมพ์	วาดสี	LOG	
จาก	เพิ่ม	ว่าง	SIN	
จุด	ภย	เว้นวรรค	SQR	
แจง	ภล	สงวน	TAN	
จอปกติ	ภลจุด	สลับแสง	&	
ช่อง	ภลต	สัมบูรณ์	+	
ซ้าย	ภลสี	สี	-	

ผ.2

ตารางรหัสอักษร

รหัสอักษร ที่ให้ไว้ในตารางนี้ เป็นรหัสอักษรที่ใช้สั่งให้พิมพ์ตัวอักษรที่ต้องการบนจอได้ โดยใช้คำสั่ง **พิมพ์ อักขระ(n)** สำหรับกรณีของอักษรไทย ต้องนำหน้าด้วย CTRL-A ซึ่งเขียนเป็น อักขระ (1) และเมื่อหมดชุดของอักษรไทยก็ต้องปิดท้ายด้วย อักขระ (1) เช่น **พิมพ์ อักขระ(1);อักขระ(33);อักขระ(34);อักขระ(35);อักขระ(1);อักขระ(33);อักขระ(34);อักขระ(35)** จะพิมพ์อักษรต่อไปนี้บนจอ **กขค!"#**

รหัส	อังกฤษ	ไทย	รหัส	อังกฤษ	ไทย	รหัส	อังกฤษ	ไทย
0	CTRL-@		20	CTRL-T		40	(	ช
1	CTRL-A		21	CTRL-U		41	)	ซ
2	CTRL-B		22	CTRL-V		42	*	ฌ
3	CTRL-C		23	CTRL-W		43	+	ญ
4	CTRL-D		24	CTRL-X		44	,	ฎ
5	CTRL-E		25	CTRL-Y		45	-	ฏ
6	CTRL-F		26	CTRL-Z		46	.	ฐ
7	CTRL-G		27	ESC		47	/	ฑ
8	CTRL-H		28			48	0	ฒ
9	CTRL-I		29	CTRL-SHIFT M		49	1	ณ
10	CTRL-J		30	CTRL-^		50	2	ด
11	CTRL-K		31			51	3	ต
12	CTRL-L		32	SPACE		52	4	ถ
13	CTRL-M		33	!	ก	53	5	ท
14	CTRL-N		34	"	ข	54	6	ธ
15	CTRL-O		35	#	ค	55	7	น
16	CTRL-P		36	$	ฆ	56	8	บ
17	CTRL-Q		37	%	ง	57	9	ป
18	CTRL-R		38	&	จ	58	:	ผ
19	CTRL-S		39	'	ฉ	59	;	ฝ

รหัส	อังกฤษ	ไทย	รหัส	อังกฤษ	ไทย	รหัส	อังกฤษ	ไทย
60	<	พ	87	W	ุ	114		ิ์
61	=	ฟ	88	X	ู	115		ี่
62	>	ภ	89	Y	ั	116		ี้
63	?	ม	90	Z	ิ	117		ี๊
64	@	ย	91	[	ี	118		ี๋
65	A	ร	92	\	ื	119		ึ่
66	B	ฤ	93	]	ึ	120		ึ้
67	C	ล	94	^	ํ	121		ึ๊
68	D	ว	95		็	122		ึ๋
69	E	ศ	96		่	123		ื่
70	F	ษ	97		้	124		ื้
71	G	ส	98		๊	125		ื๊
72	H	ห	99		๋	126		ื๋
73	I	ฬ	100		์	127		
74	J	อ	101					
75	K	ฮ	102		O่			
76	L	ฮ	103		O้			
77	M	[ ]	104		O๊			
78	N	า	105		O๋			
79	O	ำ	106		ั่			
80	P	เ	107		ั้			
81	Q	แ	108		ั๊			
82	R	โ	109		ั๋			
83	S	ใ	110		ิ่			
84	T	ไ	111		ิ้			
85	U	ๆ	112		ิ๊			
86	V	ฯ	113		ิ๋			

ผ 3

รายงานความผิดพลาด

? **เลขลำดับตัวแปลแถวผิด** (รหัสความผิดพลาด 107)

มีการเรียกใช้ตัวแปลแถว โดยอ้างเลขลำดับผิด หรือเลขลำดับเกินกว่าที่ได้กำหนดไว้ด้วยคำสั่ง **ขนาด**

? **ทำต่อไม่ได้**

มีการสั่ง **ทำต่อ** แต่โปรแกรมจบแล้ว หรือไม่มีโปรแกรม หรือมีความผิดพลาดร้ายแรงเกิดขึ้น หรือได้มีการแก้ไขโปรแกรมไปแล้ว

? **ห้ามหารด้วยศูนย์** (รหัสความผิดพลาด 133)

มีการสั่งให้หารด้วยนิพจน์ซึ่งมีค่าเป็นศูนย์

? **สูตรซับซ้อนเกินไป** (รหัสความผิดพลาด 191)

มีการใช้คำสั่ง **ถ้า (นิพจน์ของขบวนอักขระ) แล้ว** ซ้อนกันมากกว่า 2 ชั้น

? **ไม่ใช่คำสั่งตรง**

มีการใช้คำสั่ง **รับ, กำหนดสูตร, จับ** ในลักษณะคำสั่งตรง

? **ปริมาณต้องห้าม** (รหัสความผิดพลาด 53)

มีการระบุค่าที่ไม่ถูกต้อง

? **มีเพิ่มไม่มีจาก** (รหัสความผิดพลาด 0)

มีการใช้คำสั่ง **เพิ่ม** ที่ไม่คู่กับคำสั่ง **จาก**

? **หมดข้อมูลแล้ว** (รหัสความผิดพลาด 42)

มีการสั่งให้ **อ่าน** แต่ไม่มีข้อมูลแล้ว

? หน่วยความจำไม่พอ (รหัสความผิดพลาด 77)

หน่วยความจำมีไม่พอ ทั้งนี้อาจเนื่องจากโปรแกรมมีขนาดใหญ่เกินไป มีการใช้คำสั่ง **จาก ... เพิ่ม** ซ้อนกันมากกว่า 10 ชั้น มีการใช้คำสั่ง **ไปภาค... กลับ** ซ้อนกันมากกว่า 24 ชั้น มีการใช้วงเล็บซ้อนกันมากกว่า 36 ชั้น หรือมีการกำหนด **ขอบจำบน, ขอบจำล่าง** ไม่เหมาะสม

? เลขล้นหลัก (รหัสความผิดพลาด 69)

มีการใช้ตัวเลขที่มีขนาดใหญ่เกินไป หรือเล็กเกินไป

? กำหนดขนาดซ้ำ (รหัสความผิดพลาด 120)

มีการใช้คำสั่ง**ขนาด** กับ ตัวแปรแถวที่ได้รับการกำหนดขนาดเรียบร้อยแล้ว อาจเกิดจากการเรียกชื่อตัวแปร โดยไม่ได้กำหนดขนาด ซึ่งเครื่องจะกำหนดให้เอง หลังจากนั้น ถ้าใช้คำสั่ง **กำหนดขนาด** จะเป็นการกำหนดซ้ำ

? มีกลับไม่มีไปภาค (รหัสความผิดพลาด 22)

มีการใช้คำสั่ง **กลับ** ที่ไม่คู่กับคำสั่ง **ไปภาค**

? ขบวนอักขระยาวเกิน

มีการสั่งให้ต่อขบวนอักขระที่ให้ความยาวรวมกัน 255 อักษร

? ไวยากรณ์ผิด

ตรวจพบความผิดพลาดเกี่ยวกับตัวสะกด เครื่องหมายวรรคตอน ลำดับวิธีเขียนคำสั่ง หรือความผิดพลาดอื่นที่มิได้ระบุไว้

? ข้อมูลผิดประเภท (รหัสความผิดพลาด 163)

มีการใช้ตัวแปรเลข หรือนิพจน์เลขแทนที่จะเป็นขบวนอักขระ หรือกลับกัน และในกรณีที่คำสั่งกำหนดค่าตัวแปร อ้างถึงปริมาณต่างชนิดกัน

? สูตรไม่ได้กำหนด (รหัสความผิดพลาด 224)

มีการอ้างถึงสูตรซึ่งยังไม่ได้กำหนด

? คำสั่งไม่ปรากฏ (รหัสความผิดพลาด 90)

มีการสั่งให้ไปหมายเลขบรรทัดที่ไม่มีในโปรแกรม

หอสมุดแห่งชาติ
NATIONAL LIBRARY